DESIGN AND DEVELOPMENT OF MICROCONTROLLER SYSTEM, EMBEDDED CIRCUITS AND APPLICATIONS

Thesis
Submitted to the Saurashtra University, Rajkot
For
The Degree of Doctor of Philosophy
Faculty—Science
Subject—Electronics

By
Anand Atalbihari Bhaskar
Lecturer & Research Scholar
Department of Electronics
Saurashtra University
Rajkot-360005
Gujarat-INDIA

Research Supervisor
Dr. H. N. Pandya
Head
Department of Electronics
Saurashtra University
Rajkot-360005
Gujarat-INDIA.

April 2009

Registration No. 3272 & Date: 29/07/2005.

This one is for

My Parents

Shri Atalbihari Sumerlal Bhaskar

Smt. Saroj A Bhaskar

And

My Elder Sister & Younger Brother

Heena Didi and Sachin

With all my love.

Statement Under O.Ph.D. Of Saurashtra University

The content of this thesis is my own work carried out under the supervision of Dr. H. N. Pandya and leads to some contribution in Electronics supported by necessary references.

(A. A. Bhaskar)

This is to certify that the present work submitted by Mr. Anand A Bhaskar for the Ph.D. degree of Saurashtra University, Rajkot has been the result of about four years work under my supervision and is a valuable contribution in the field of Electronics.

Dr. H. N. Pandya
Head
Department of Electronics
Saurashtra University,
Rajkot-360005,
Gujarat-INDIA.

Acknowledgement

The future belongs to those who believe in the beauty of their dreams. I saw a dream five years back, and now when its time to come true I would like to express my gratitude and heartily feeling in words to those who make it look easier for me. Very firstly I would like to thanks my parents Shri. A. A. Bhaskar and Smt S. A. Bhaskar for having faith on me and supporting me in every phase of my life, they really stayed in the worst case scenario and helping me out in every trouble that I was facing throughout my life. And a special thanks to my sister Heena and brother Sachin who were always boosting me every time.

I have my immense pleasure in submitting the presented research work which is the result of my humble effort to carry out some piece of work in the field of Electronics which would not have been possible to get its present shape without the support and hearty wishes of my ideal, mentor and guide Dr. H. N. Pandya. His inspiring guidance, continuous encouragement, thoughtful discussion and untiring supervision throughout my whole Ph.D. work are great help for getting this milestone in my life. His guidance has been so sublime as to render it possible for me to build the work to its present shape. Dr. H. N. Pandya, has been kind enough to accept me as a student for the research work and inspired me with profound love so that I succeed in doing what was expected from me. Not only him but his family too, I can't express my gratitude in words for Mrs. Kiran H Pandya and Ms. Sweta H Pandya.

Dr. Darshan G Vyas and Dr. Maulin N Nanavati are the two key persons who really helped me in my early days of my Ph.D. work. I learnt a lot from the experience of these two. I am also thankful to Dr. Kapil Bhatt for visiting the laboratory and proving his suggestions for the future research projects.

I am thankful to Mr. H. G. Chirutkar for starting the journey of research with me and for making kind adjustments in the research laboratory. Without his helping hand, I definitely would not have reached this goal.

I express my sincere gratitude and heart felt thanks to Mr. Sandeep Pandey for providing a better company during the research as one of the motivator and friend.

I also thank Ms. Seema J Oza for undertaking one of my dream projects in the field of robotics as her thesis work.

I extend my sincere thanks to Department of Electronics, Saurashtra University, Rajkot staff members Mr. Manish R. Pandya, Mr. H.G. Pandya and Dhirubhai for doing and providing valuable facilities for the present work in my work place and office.

I am also thankful to the Saurashtra University for proving me a good hostel facility during the first two years of my research work and also to Mr. Yagnesh Pandya's family for proving kind support from last two years. Especially Divya who is only two years of age and make me forcefully read EFORU with him every night.

Finally I am thankful to the almighty GOD for waking me up every morning to see my dreams coming true with open eyes.

CONTENTS

Section - I: Aims and tools.

Section - II: Design of the microcontroller based system.

Section - I: Aims and tools.

Section 1: Aim and Tools.

Chapter 1 Introduction

Each day, our lives become more dependent on 'embedded systems', digital information technology that is embedded in our environment. Microcontrollers are the most used devices in embedded systems. Microcontrollers, as the name suggests, are small controllers. They are like single chip computers that are often embedded into other systems to function as processing/controlling unit. The key features of microcontrollers include:

- High Integration of Functionality: Microcontrollers sometimes are called single-chip computers because they have on-chip memory and I/O circuitry and other circuitries that enable them to function as small standalone computers without other supporting circuitry.
- Field Programmability, Flexibility: Microcontrollers often use EEPROM or EPROM as their storage device to allow field programmability so they are flexible to use. Once the program is tested to be correct then large quantities of microcontrollers can be programmed to be used in embedded systems.
- Easy to Use: Assembly language is often used in microcontrollers and since they usually follow RISC architecture, the instruction set is small. The development package of microcontrollers often includes an assembler, a simulator, a programmer to "burn" the chip and a demonstration board. Some packages include a high level language compiler such as a C compiler and more sophisticated libraries.

Modern technology demands from any engineer, a basic microcontroller or microprocessor knowledge. The basic difference between them is that microprocessors can be configured for the amount of memory and the input/output system used. The microcontroller has all the computing system (I/O system and memory) built in it. Designer’s judgment determines which one should be used. Unlike desktop computer, microcontrollers interact with other machines rather than humans. A microcontroller might be used to measure the temperature of your toast at breakfast and when the temperature reaches a predetermined measure, the toaster could be turned off. A microcontroller could also be used to count the number of customers entering the ball park through a turnstile thereby keeping track of ticket sales. The uses for these small versatile devices are diverse. Perhaps one can imagine a microcontroller application that will improve a product or decrease the time required to complete a process.

The emphasis of this work will be in the one of the better development cycle and embedded system design. Almost all important microcontroller parts such as the memory, the I/O system; it’s interfacing with the other devices and applications are covered in the present work.

1.1 Objectives of the present work

Following objectives are achieved during the research work as it was discussed in the RDC meeting conducted by Saurashtra University during year 2005.

- Design and Development of microcontroller based systems helpful in the dissemination of technical and scientific knowledge.
- Design and Development of applications based on the day to day microcontroller based gadgets.
- Design and Development of embedded systems.

For the Design and Development of microcontroller based systems MCS51 family of microcontrollers are targeted and for Design and Development of applications and embedded systems both MCS51 and ARM family is used. One of the easy and fast prototyping development cycles for the microcontroller system is achieved in the presented work. Along with that many useful and standalone embedded circuits and systems were designed.

Work Organization

Work done is presented in four sections. In the first section “Aim and Tools” basic concepts of the microcontrollers utilized in the work is provided. Basic MCS51 family architecture along with the programmer’s model is presented in the chapter 2. The ARM processor architecture and programmers model is discussed in the chapter 3. The section two “Design of the microcontroller based system” provides the designing aspect of the embedded microcontroller based systems. Chapter 4 is dedicated to the experimental board design for MCS51 family of microcontroller. The debugging tool is required for the full system development cycle for the embedded system design and is presented in chapter 5. For the embedded system design MCS51 family is not the only option, there are many other higher end microcontrollers are available in the market which are having many useful features embedded in it. Next perspective for the further research is focused from the basic ARM7 board study and design and is given in the chapter 6. The section three “Design of microcontroller based interfacing modules and application” comprises of the chapters 7-10. In these chapters 7-10 detailed description of the microcontroller interfacing and its applications are provided. For the embedded circuit and system design, section four “Design of Microcontroller based embedded circuits and system” is dedicated. Chapter 11 shows the detailed description of the MCS51 based embedded system design by an application design of RF test kit. ARM7 based embedded system developments detailed description with an application RFID test kit is provided in the chapter 12.

1.2 Tools Utilized

Tools utilized in the present work are categorized as follows:

- Hardware Tools – For the microcontroller based embedded system design various hardware tools are utilized which are available the Department of Electronics, Saurashtra University, Rajkot. Following is the list of the utilized tools.
    1. Microcontroller Test Kits: Dyna51, MCS51 (8051 based trainer kits) and ARM7 educational kit (ARM architecture trainer kits).
    2. Workstations: PCB fabrication tools, PC, and Equipped electronics workbench.
    3. Debugging Tools: MCS51 PCICE (In-Circuit Emulator for 8051/52 microcontroller based systems)
    4. Miscellaneous: MCS51 and ARM family of microcontrollers, LCD module, 40-key Matrix Keyboard, 5x7 LED matrix, RFID module, RF module, cable-connectors and supported electronic devices.
- Software Tools – All the software tools must be installed in the PC workstation, they are listed below.
    1. Assembles & Compilers: ML_ASM51, ASM51, Keil µvision3
    2. Simulators & Emulators: Emily (demo version for MCS51 microcontrollers), PICE emulator (Full version by phyton for MCS51), Keil µvision3.
    3. Philips Flash Utility Software: WinISP, Flash Magic, LPC2000 Flash Utility V2.2.3.
    4. HJTAG Software for ARM family.

Chapter 2 MCS51 Family Architecture

Introduction

The 8051 is the original member of the MCS-51 family, and is the core for all MCS-51 devices. The features of the 8051 core are as listed below:

- 8-bit CPU optimized for control applications
- Extensive Boolean processing (Single-bit logic) capabilities
- 64K Program Memory address space
- 64K Data Memory address space
- 4K bytes of on-chip Program Memory
- 128 bytes of on-chip Data RAM
- 32 bidirectional and individually addressable I/0 lines
- Two 16-bit timer/counters
- Full duplex UART
- 6-source/5-vector interrupt structure with two priority levels
- On-chip clock oscillator

The 8051 processor architecture is Harvard-based with external memory read/write capabilities built in as part of the architecture. Figure 2-1 shows the basic 8051 architecture.

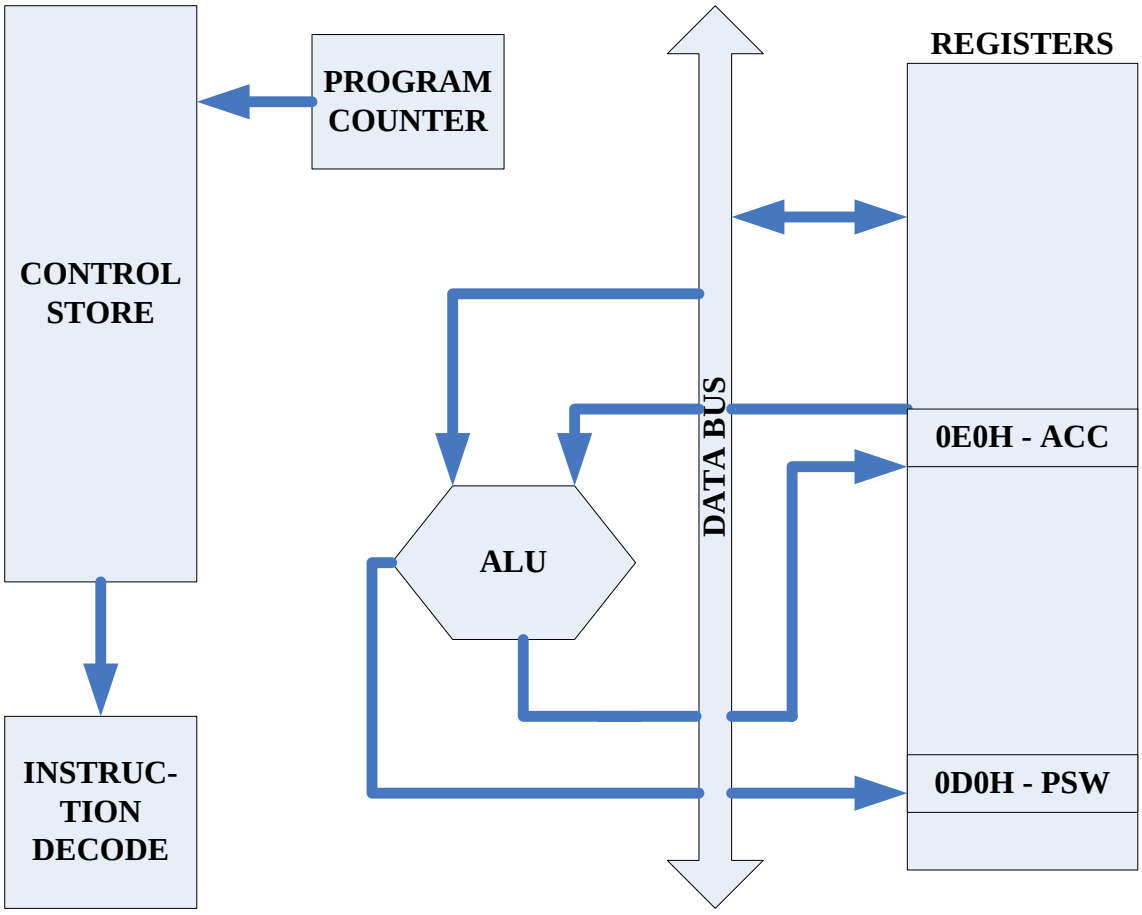


Figure 2-1: Basic 8051 Architecture for the MCS51 family of microcontrollers.

2.1 Overview of the MCS51 family microcontroller

Despite its relatively old age, the 8051 is one of the most popular microcontrollers in use today. Many derivative microcontrollers have since been developed that are based on and compatible with the 8051. The 8051 is an 8-bit processor, meaning that the CPU can work on only 8 bits of data at a time. Data larger than 8-bits has to be broken into 8-bit pieces to be processed by the CPU. The working of the microcontroller can be understood easily by knowing the internal architecture of the microcontroller. This is shown in the 8051 microcontroller block diagram.

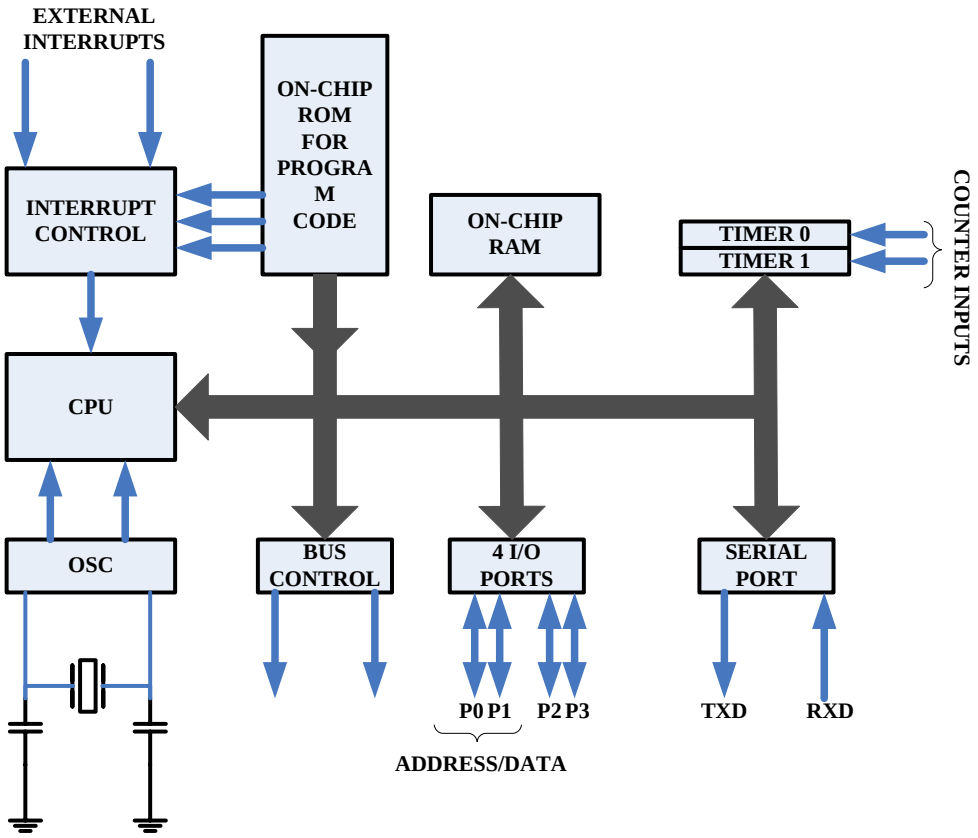


Figure 2-2: The 8051 microcontroller block diagram.

The 8051 family members like 8751, 89C51, 89C52, DS89C4x0, P89C51RD2 come in different packages. They all have 40 pins that are dedicated to various functions such as I/O, RD, WR, address, data, and interrupts. The 8051 pin diagram is shown in Figure 2-3.

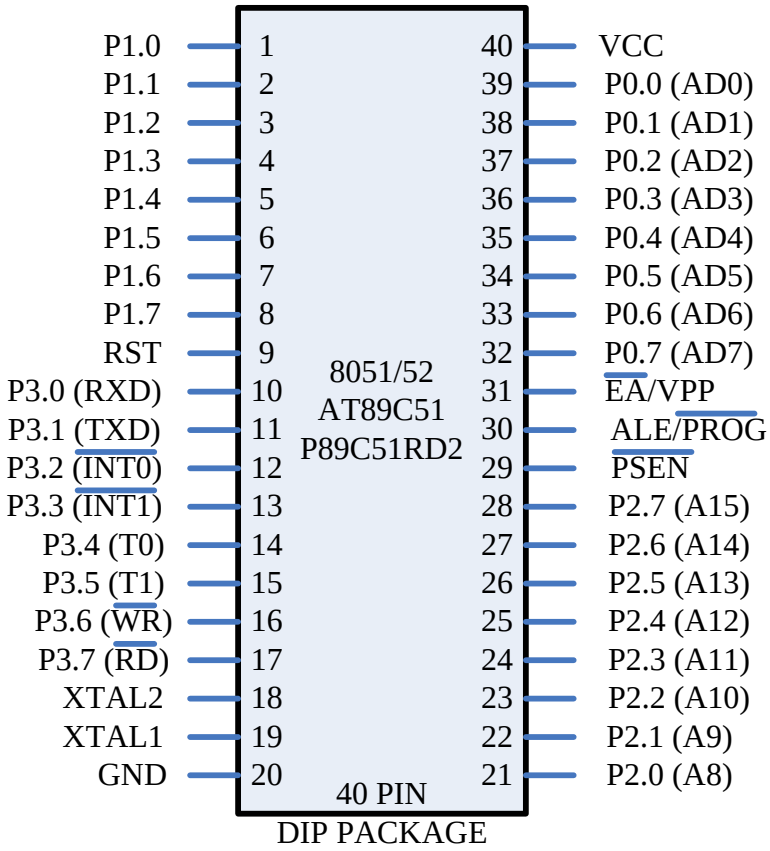


Figure 2-3: 8051 microcontroller pin diagram.

PIN(s)	FUNCTION
1-8	Port 1. The bi-directional pins on this port may be used for input and output. Each pin may be individually controlled.
9	The Reset. When this pin is held at logic 0, the chip will run normally. If, while the oscillator is running, this pin is held at logic 1 for two (or more) machine cycles, the microcontroller will be reset.
10-17	Port 3. Another bi-directional input port. Each pin on this port also serves an additional function. Pin 10 and Pin 11 are used to receive and transmit serial data using the RS232 protocol respectively. Pin 12 and Pin 13 are used to process interrupt inputs. Pin 14 and Pin 15 have alternative functions associated with Timer 0 and Timer 1. Pin 16 and Pin 17 are used when working with external memory.
18-19	These pins are used to connect an external crystal, ceramic resonator or oscillator module to the microcontroller.
20	Vss. This is the ground pin.
21-28	Port 2. Another bi-directional input port. Same operation as port 1.
29	Program Store Enable (PSEN) is used to control access to external CODE memory, if used.
30	Address Latch Enable (ALE) is used when working with external memory.
31	External Access (EA). To execute code from internal memory this pin must be connected to Vcc. To execute code from external memory, this pin must be connected to ground.
32-39	Port 0. Another bi-directional input port. Same operation as port 1.

But note that unlike Port 1, Port 2 and Port 3 this port does not have internal pull-up resistors. Also used when working with external memory for addressing.

40 Vcc. This is the 5V pin.

The microcontrollers can be programmed by means of microcontroller programmers which are easily available in the market now days as well as by some flash utility software using the parallel port or serial port of the PC.
Two flavors of 8051 are discussed below which are used in the present work.

- **AT89C51 from Atmel Corporation**

The AT89C51 is a popular and inexpensive chip used in many small projects. "AT" is for the company name Atmel and "C" before the 51 stands for CMOS, which has low power consumption. It has 4K bytes of flash ROM and comes in DIP, QFP and LLC package. In the present work only 40 pin DIP packages are used for the system as well as for application design. The AT89C51 provides the following standard features:
1. 4K bytes of Flash
2. 128 bytes of RAM
3. 32 I/O lines
4. Two 16-bit timer/counters
5. A five vector two-level interrupt architecture
6. A full duplex serial port
7. On-chip oscillator and clock circuitry.

In addition, the AT89C51 is designed with static logic for operation down to zero frequency and supports two software selectable power saving modes. The Idle Mode stops the CPU while allowing the RAM, timer/counters, serial port and interrupt system to continue functioning. The Power-down Mode saves the RAM contents but freezes the oscillator disabling all other chip functions until the next hardware reset.

- **P89CRD2XX from Philips Corporation**

The P89C51RD2 device contains a non-volatile 64kB Flash program memory that is both parallel programmable and serial In-System and In-Application Programmable. In-System Programming (ISP) allows the user to download new code while the microcontroller sits in the application. In-Application Programming (IAP) means that the microcontroller fetches new program code and reprograms itself while in the system. This allows for remote programming over a modem link. A default serial loader (boot loader) program in ROM allows serial In-System programming of the Flash memory via the UART without the need for a loader in the Flash code. For In-Application Programming, the user program erases and reprograms the Flash memory by use of standard routines contained in ROM. This device executes one machine cycle in 6 clock cycles, hence providing twice the speed of a conventional 80C51. An OTP configuration bit lets the user select conventional 12 clock timing if desired. This device is a Single-Chip 8-Bit Microcontroller manufactured in advanced CMOS process and is a derivative of the MCS51 microcontroller family. The instruction set is 100% compatible

with the MCS51 instruction set. The device also has four 8-bit I/O ports, three 16-bit timer/event counters, a multi-source, and four-priority-level, nested interrupt structure, an enhanced UART and on-chip oscillator and timing circuits. The added features of the P89C51RD2 make it a powerful microcontroller for applications that require pulse width modulation, high-speed I/O and up/down counting capabilities such as motor control. The P89C51RD2 provides following standard features:

1. 80C51 Central Processing Unit
2. On-chip Flash Program Memory with In-System Programming (ISP) and In-application Programming (IAP) capability
3. Boot ROM contains low level Flash programming routines for downloading via the UART
4. Can be programmed by the end-user application (IAP)
5. 6 clocks per machine cycle operation (standard), 12 clocks per machine cycle operation (optional)
6. Speed up to 20 MHz with 6 clock cycles per machine cycle (40 MHz equivalent performance); up to 33 MHz with 12 clocks per machine cycle
7. Fully static operation
8. RAM expandable externally to 64 kB
9. 4 level priority interrupt
10. 7 interrupt sources
11. Four 8-bit I/O ports
12. Full-duplex enhanced UART
13. Power control modes
14. Second DPTR register
15. Asynchronous port reset
16. Programmable Counter Array (PCA)

2.2 The MCS51 programmer's model

One thing that scares many people off from starting to work with microcontrollers is the perception that the devices are difficult to load with an application and require expensive programmers. This is generally not true for most devices, and programmers capable of working with a variety of parts are available for quite reasonable costs. In the present work, for 8051 programming two perspectives are used. Initially I have started programming the microcontrollers by using universal programmer LabTool available in the market. But later found that the algorithms written to program the microcontroller are not quite complex. So I went for the ISP (In-System Programming) by using the flash utility software to program the microcontroller, which is to be serially connected to the PC at the time of programming. Figure 2-4 shows the programming connections for the 8051 family of microcontrollers.

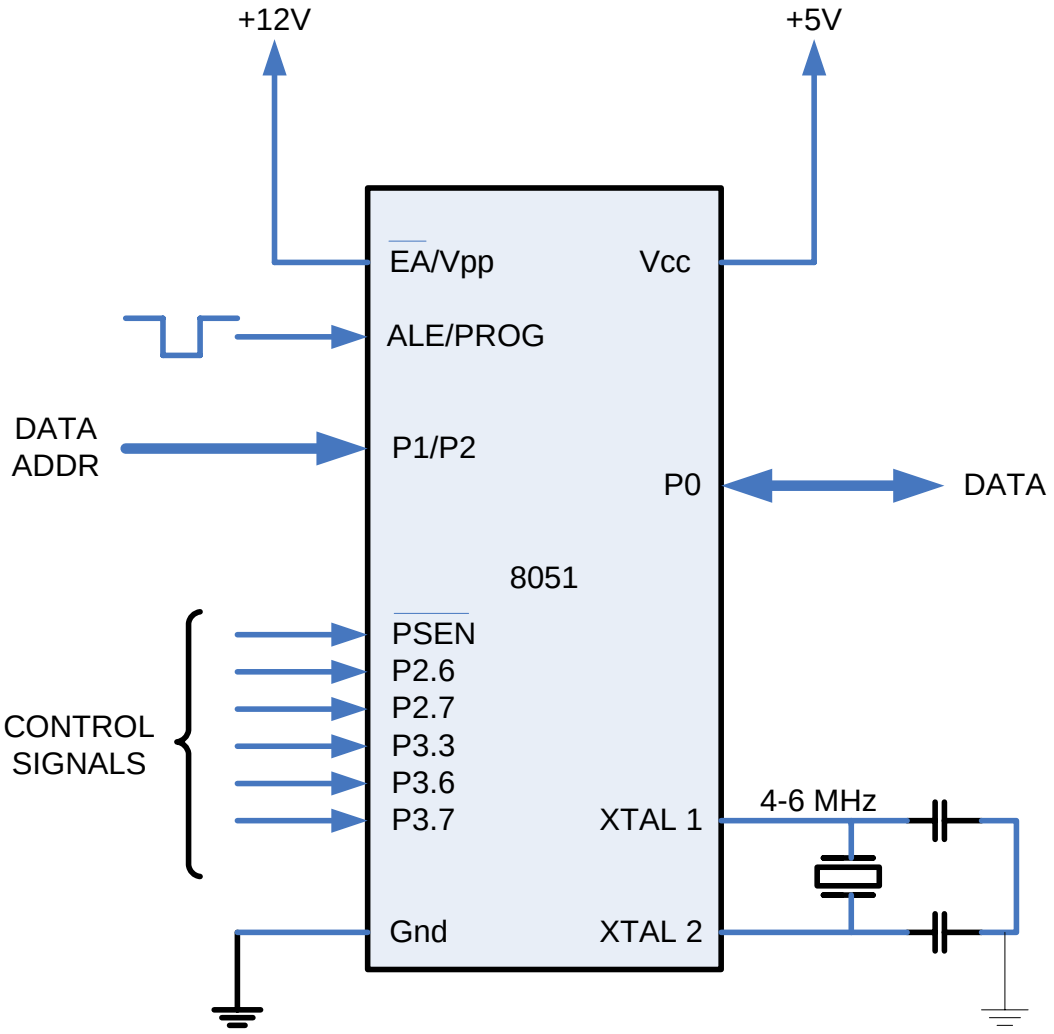


Figure 2-4: 8051 programming connections.

The internal organization of the microcontroller in terms of memory space and registers is an important perspective for the programmer to write software for any problem statement. The programming model for the 8051 is shown in the Figure 2-5. The programming model shows the 8051 in terms of 8-bit and 16-bit registers and also 8-bit memory locations. The model seems complex by the number of special purpose registers that must be present to make a microcomputer a microcontroller. Almost all of the registers have specific functions.

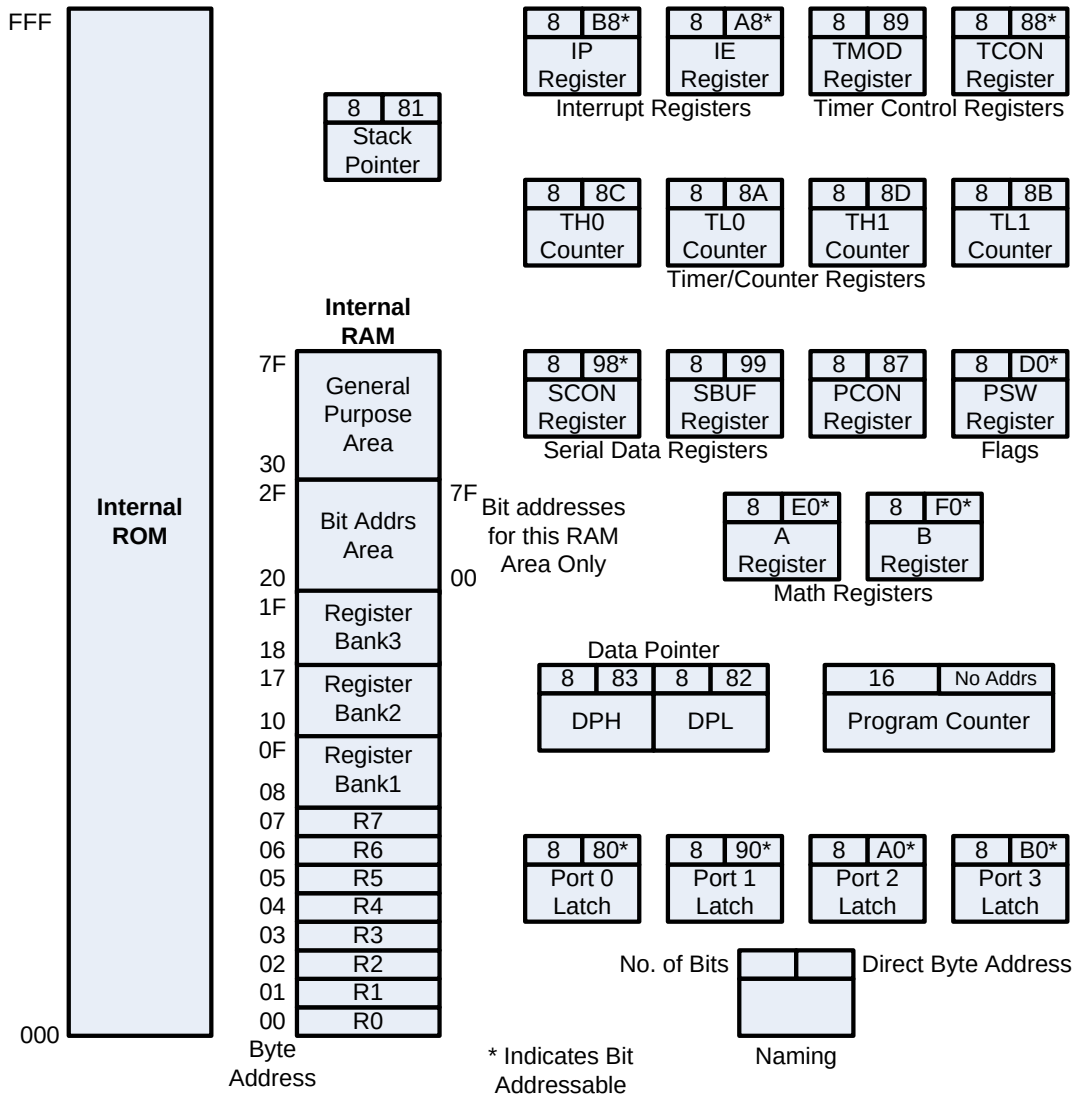


Figure 2-5: Programming Model for 8051 microcontroller.

The detailing of the instruction set is provided in the Appendix A.

Chapter 3 The ARM processor Architecture

Introduction

The ARM processor is a Reduced Instruction Set Computer (RISC). In this chapter the RISC ideas are mainly focused to shape the ARM processors. The ARM was originally developed at Acorn Computers Limited of Cambridge. The principle features of the ARM architecture are presented here in the overview form.

3.1 Overview of the Acorn RISC Machine

Initially ARM stands for Acorn RISC Machine but later more advancement are done in the processor architecture and acronym expansion is for Advanced RISC Machine.

At the time the first ARM chip was designed, the only examples of RISC architecture were the Berkeley RISC I and II and the Standford MIPS, although some earlier machines such as the Digital PDP-8, the Cray-1 and the IBM 801, which predated the RISC concept, shared many of the characteristics which later came to be associated with RISCs.

The ARM architecture incorporated a number of features from the Berkeley RISC design, but a number of other features were rejected. Those that were used are:

1. A load store architecture
2. Fixed length 32-bit instructions
3. 3-address instruction formats

The features that were employed on the Berkeley RISC designs but were rejected by the ARM designers are:

1. Register windows
2. Delayed branches
3. Single cycle execution of all instructions

The overriding concern of the original ARM design team was the need to keep the design simple. The simplicity of the ARM may be more apparent in the hardware organization and implementation than it is in the instruction set architecture. From the programmer's perspective it is perhaps visible as conservatism in the ARM instruction set design which, while accepting the fundamental precepts of the RISC approach, is less radical than many subsequent RISC designs. The combination of the simple hardware with an instruction set that is grounded in RISC ideas but retains a few key CISC features, and thereby achieves a significantly better code density than a pure RISC, has given the ARM its power efficiency and its small core size.

Both hardware and software support for the ARM processor is now available. The ARM is supported by a toolkit which includes an instruction set emulator hardware modeling and software testing and benchmarking, an assembler, C and C++ compilers, a linker and a symbolic debugger.

3.2 The ARM Architecture overview

- Figure 3-1 shows the internal structure of the ARM processor.

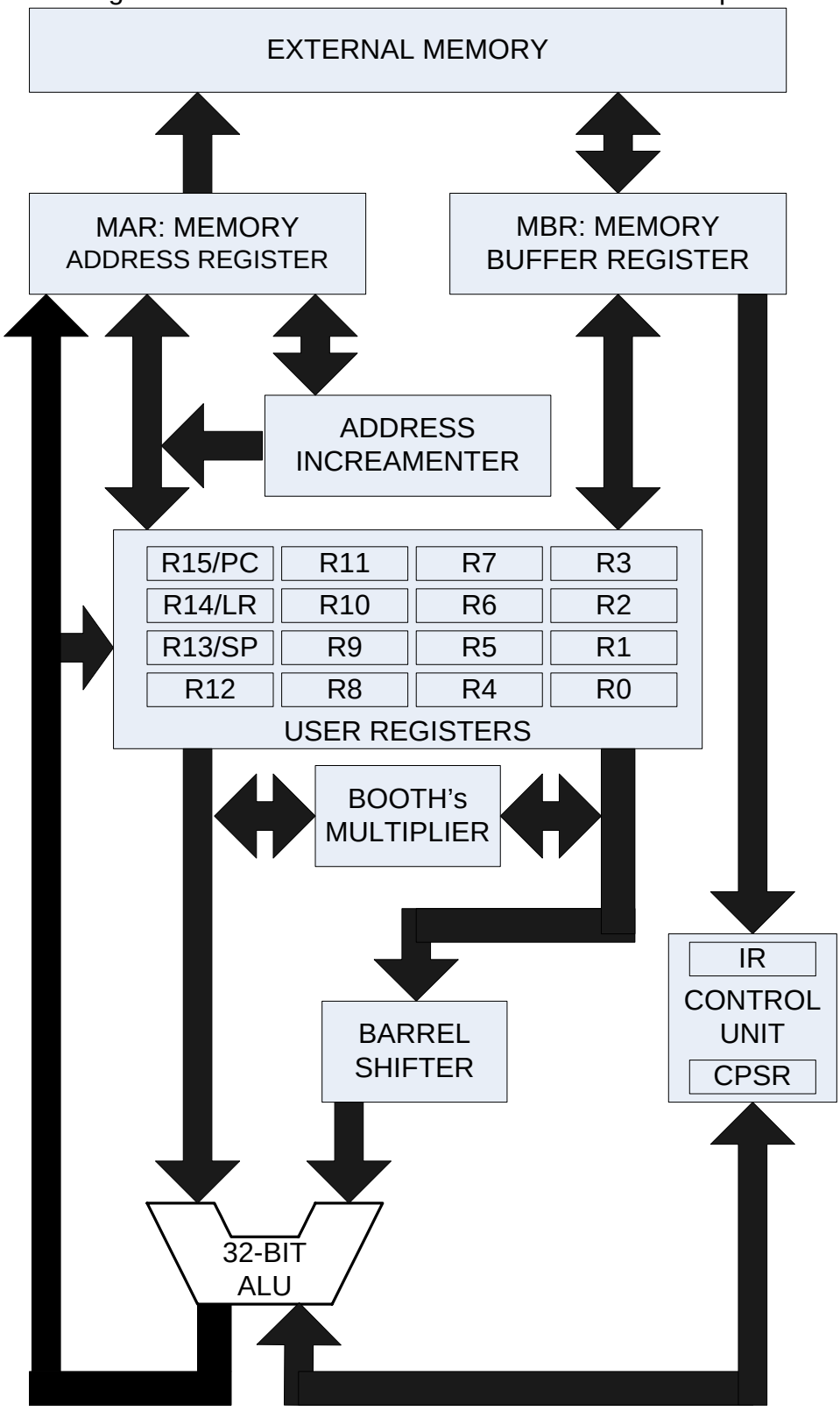


Figure 3-1: ARM block diagram.

The ARM is a Reduced Instruction Set Computer (RISC) system and indicates the attributes typical to that type of system as listed below:

- A large array of uniform registers.
- A load/store model of data-processing where operations can only operate on registers and not directly on memory. This requires that all data be loaded into registers before an operation can be performed, the result can then be used for further processing or stored back into memory.
- A small number of addressing modes with all load/store addresses begin determined from registers and instruction fields only.
- A uniform fixed length instruction (32 bit).

In addition to these features of a RISC system the ARM provides a number of additional features.

- Separate Arithmetic Logic Unit (ALU) and shifter giving additional control over data processing to maximize execution speed.
- Auto increment and Auto decrement addressing modes to improve the operation of program loops.
- Conditional execution of instructions to reduce pipeline flushing and thus increase execution speed.

The ARM supports the seven processor modes shown in Table 3-1.

Processor	Mode	Description
User	usr	Normal program execution mode
FIQ	fiq	Fast Interrupt for high speed data transfer
IRQ	Irq	Used for general purpose interrupt handling
Supervisor	svc	A protected mode for the operating system
Abort	abt	Implements virtual memory and/or memory protection
Undefined	und	Supports software emulation of hardware coprocessors
System	sys	Runs privileged operating system task

Table 3-1: ARM processor modes

Mode changes can be made under software control, or can be caused by external interrupts or exception processing. Most application programs execute in user mode. While the processor is in user mode, the program being executed is unable to access some protected system resources or to change mode, other than by causing an exception to occur. This allows a suitable written operating system to control the use of system resources.
The modes other than User mode are known as privileged modes. They have full access to system resources and can change mode freely. Five of them are known as exception modes: FIQ (Fast Interrupt), IRQ (Interrupt), Supervisor, Abort and Undefined. These are entered when specific exceptions are occurred. Each of them has some additional registers to avoid corrupting User mode state when the exception occurs. The remaining mode is the System mode; it is not entered by any exception and has exactly the same registers available as user mode. However, it is a privileged mode and is therefore not subject to the user mode restrictions. It is intended for the use by operating system tasks which need access to system resources, but wish to avoid using the additional registers associated with the exception modes. Avoiding such

use ensures that the task state is not corrupted by the occurrence of any exception.

3.3 The ARM7 programmer's model

A processor's instruction set defines the operations that the programmer can use to change the state of the system incorporating the processor. This state usually comprises the values of the data items in the processor's visible registers and the system's memory. Each instruction can be viewed as performing a defined transformation from the state before the instruction is executed to the state after it has completed. The visible registers in an ARM processor are shown in Figure 3-2.

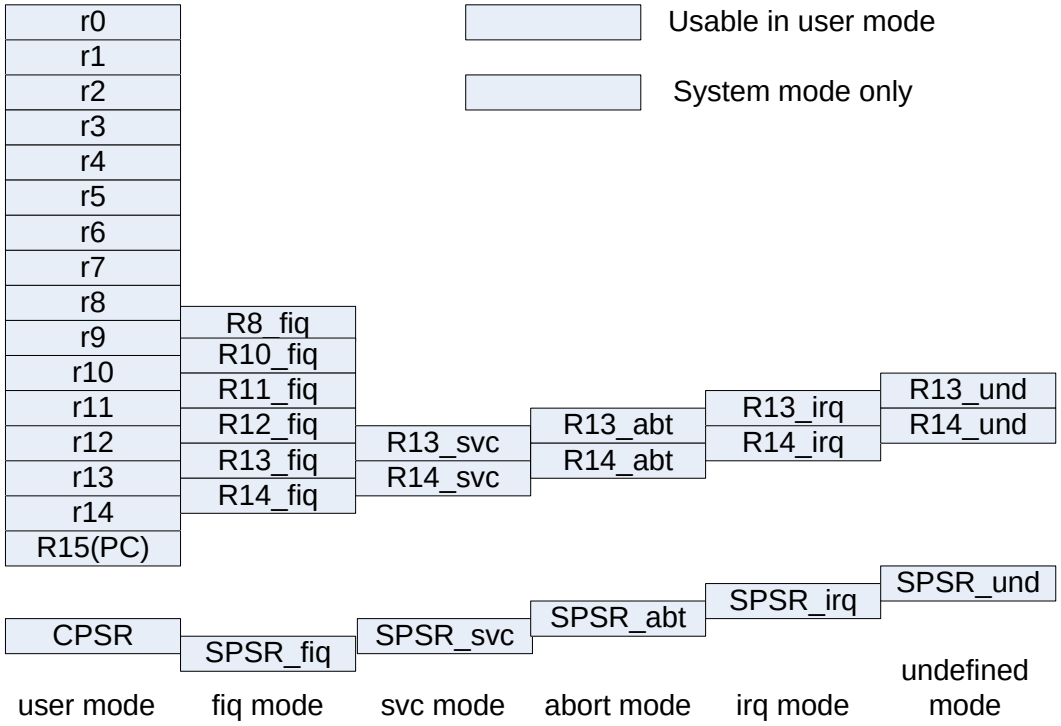


Figure 3-2: ARM's visible registers.

When writing user-level programs, only the 15 general-purpose 32-bit registers (r0 to r14), the program counter (r15) and the current program status register (CPSR) need to be considered. The remaining registers are used only for system-level programming and for handling exceptions.

The CPSR

The CPSR is used in user-level programs to store the condition code bits. Figure 3-3 shows the ARM CPSR format.

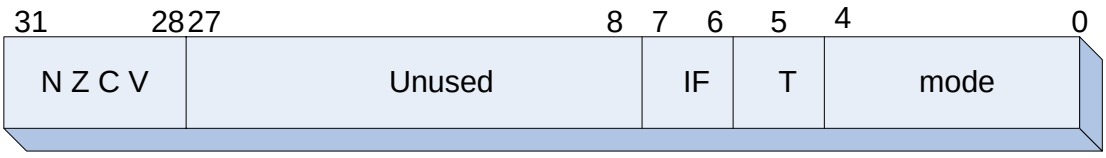


Figure 3-3: ARM CPSR format.

The bits at the bottom of the register control the processor mode, instruction set and interrupt enables and are protected from change by the user-level program. The condition code flags are in the top four bits of the register and have the following meaning.

- N: Negative; the last ALU operation which changed the flags produced a negative result.
- Z: Zero; the last ALU operation which changed the flags generated a zero result.
- C: Carry; the last ALU operation which changed the flags generated a carry-out, either as a result of an arithmetic operation in the ALU or from the shifter.
- V: overflow; the last arithmetic ALU operation which changed the flags generated an overflow into the sign bit.

The memory organization

The memory organization is shown in the figure 3-4.

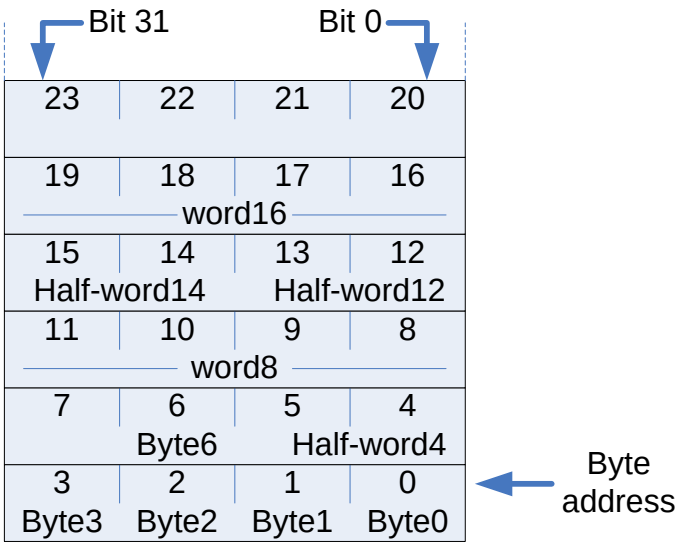


Figure 3-4: ARM memory organization.

This shows a small area of memory where each byte location has a unique number. A byte may occupy any of these locations, and few examples are shown in the figure.

Load store architecture

The instruction set will only process values which are in registers, and will always place the results of such processing into memory. Moreover all ARM instructions fall into one of the following three categories:

1. Data processing instructions
2. Data transfer instructions
3. Control flow instructions

The I/O system

The ARM handles I/O (input/output) peripherals as memory mapped devices with interrupt support. The internal registers in these devices appear as addressable locations within the ARM's memory map and map be read and written using the same instructions as any other memory locations. Peripherals may attract the processor's attention by making an interrupt request using either the normal interrupt (IRQ) or the fast interrupt (FIQ) input. Some systems may include direct memory access (DMA) hardware external to the processor to handle high bandwidth I/O traffic.

ARM7 Family

The ARM7 core has a Von Neumann style architecture, where both data and instructions use the same bus. The core has a three-stage pipeline and executes the architecture ARMv4T instruction set. The present work is developed around the ARM7TDMI. The ARM7TDMI was the first of a new range of processors introduces in 1995 by ARM. It is currently a very popular core and is used in many 32-bit embedded processors. It provides a very good performance to power ratio. The ARM7TDMI processor core has been licensed by many of the top semiconductor companies around the world and is the first core to include the Thumb instruction set, a fast multiply instruction, and the Embedded ICE debug technology.

Section - II: Design of the microcontroller based system.

Section 2: Design of the microcontroller based system.

Chapter 4 Experimental Board for MCS51 Family of Microcontroller

4.1 Introduction

SU51 MB is a microcontroller trainer designed and developed as part of Ph.D. work which is based on the popular 8051 core. It can be used as a flexible instructional aid in academic institution. This is a general purpose board designed as a development tool; this board has a facility to download HEX file into the on-chip flash code memory of the micro-controller, without the need of removing the chip from the socket. Now a day the microcontrollers are extensively used for embedded and real time applications. It is easy to use, and design to be a general purpose development board for single chip micro-controlled unit applications that may be used as a development tool in R&D labs in universities and industries.

SU51 MB is versatile, general purpose controller using the popular Philips P89C51RD2 microcontroller that has 64Kbytes of on-chip program memory. It is designed for use in applications ranging from industrial control to robotics.

4.2 Microcontroller Development Boards

In system based on an AT89C51/52 type microcontroller, one needs a ROM/Flash burner to burn the program into the microcontroller. For the AT89C51, the ROM burner can erase the flash ROM in addition to burning a program into it. In case of the 8751, one also needs an EPROM erasure tool since it uses UV-EPROM. To burn the 8751, one needs to erase its contents first, which takes approximately 20 minutes for UV-EPROM.

The P89C51RD2 chip from Philips semiconductor is an 8051 type microcontroller with on-chip flash ROM. It also has a built-in loader allowing it to download programs into the chip via the serial port, therefore eliminating any need for an external ROM burner. This important feature makes the P89C51RD2 chip an ideal candidate for 8051-board for further experimentation.

Key Features of the P89c51RD2

The following are some of the key features of the P89C51RD2 chip taken from Philips semiconductor web site (http://www.semiconductors.philips.com).

- 80C51 CPU
- 64K bytes of flash (Code Memory).
- 1K bytes of RAM (Data Memory).
- 1K Boot ROM contains low level flash programming routines for downloading via the UART.
- Watch Dog Timer.
- Programmable Counter Array (PCA) on Port1 i.e. PWM and Capture & Compare.

- Four 8-bit I/O ports.
- 4 level priority interrupts.

Key Features of the SU51 MB (Saurashtra University MCS51 Microcontroller Board)

- Philips P89C51RD2 operating at 11.0592 MHz crystal.
- WINISP programming tools.
- RS 232 port for direct connection to PC's serial port.
- ISP support for on-chip flash programming.
- All 32 I/O port lines are terminated on 4 different 8-pin connectors.
- DIP switch setting for mode selection (Program or Execution).
- On-Board reset switch.
- Power supply: +5V (not on board) and Power indicator LED.

4.3 Block Diagram of the SU51MB

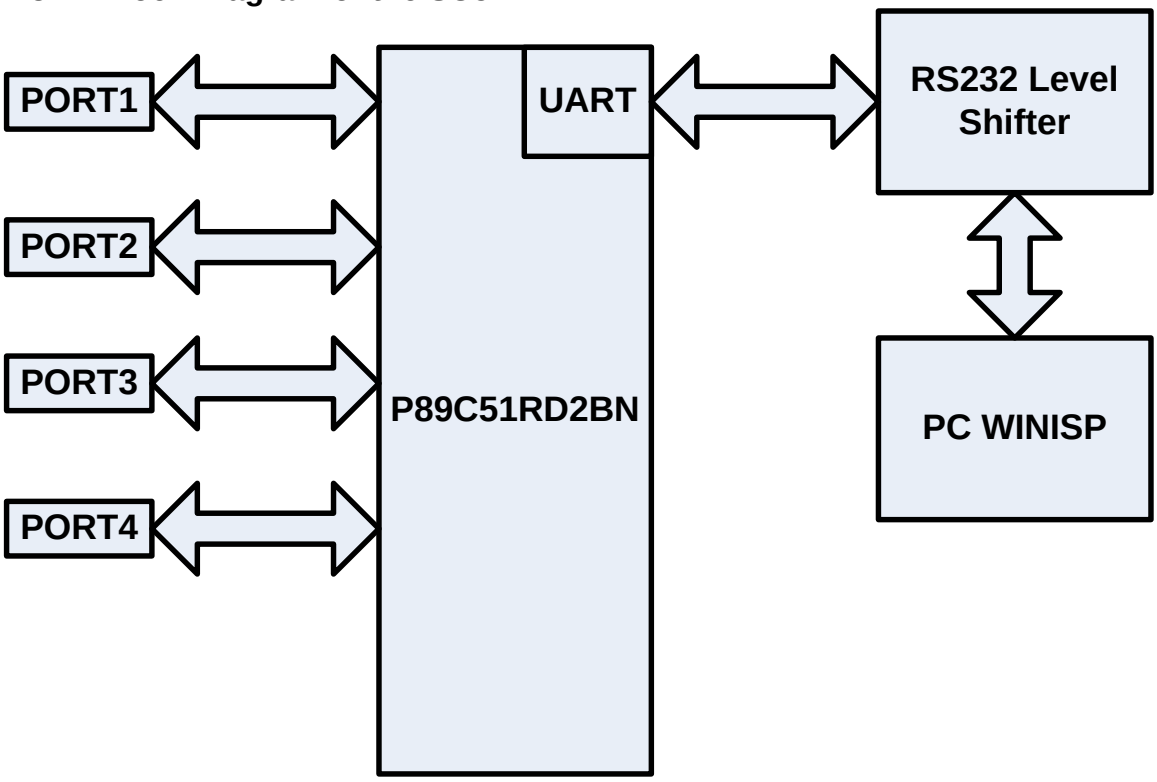


Figure 4-1: Block diagram of SU51 MB.

SU51 MB can be used as a base of any microcontroller based development system. It provides the platform where user can program and execute the programs without removing the chip from the socket. All the port pins are provided to the user for any typical application. For more details of the chip please refer the data sheet of P89C51RDBN from Philips Semiconductors. Figure 4-1 shows the block diagram of the SU51MB.

4.4 SU51MB Connections

We selected the P89C51RD2 for an 8051 experiment board because it is inexpensive but powerful, and one can easily wire-wrap it to be used at work and home. Figure 4-2 shows the schematic of SU51 MB.

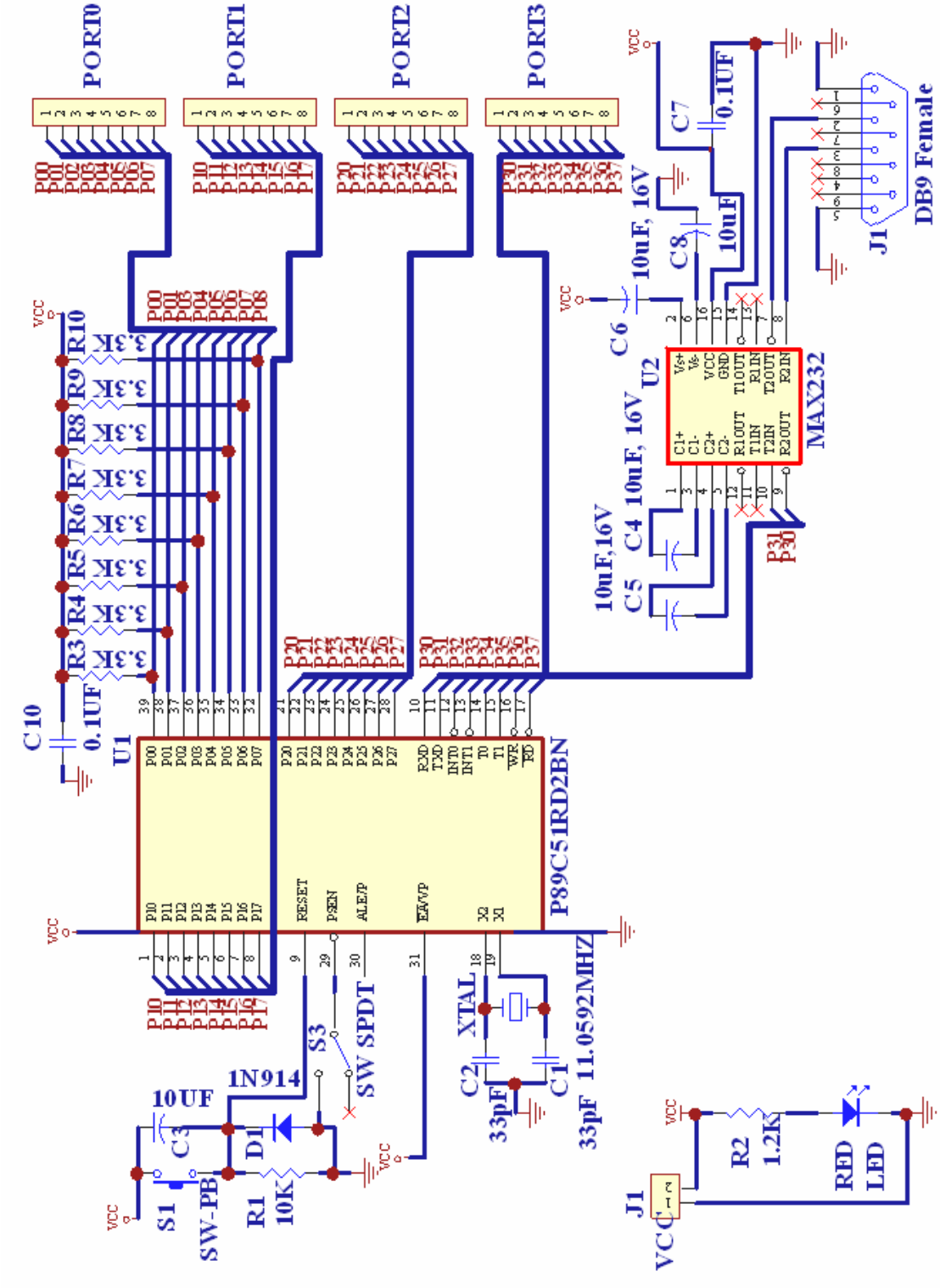


Figure 4-2: Circuit diagram of the SU51MB.

Using the P89C51RD2 for development is more advantageous than using the 89C51 or 8751 system for the following major reasons.

- Using the P89C51RD2 for an 8051 microcontroller allows one to program the chip without any need for a ROM burner. Because not everyone has access to a ROM burner, the P89C51RD2 is an ideal home development system. The advantage of the P89C51RD2 is that it can be programmed via the COM port of a PC while it is in the system. Contrast this with the 89C51 system in which one must remove the chip, program it, and install it back in the system every time one want to change the program contents of the on-chip ROM. This result in a much longer development time for the 89C51 system compared with the P89C51RD2 system.
- Once the IC is programmed user can use the board as general purpose development board for any application interface.

4.5 Hardware Details of SU51MB

CPU

SU51 MB is designed around P89C51RD2BN microcontroller. The board has power on reset circuit, to reset the microcontroller press the PB-Switch provided onboard. CPU is working at 11.0592 MHz clock frequency.

On-Chip Memory Map

The P89C51RD2BN provides on-chip flash program memory of 64K Bytes; range from 0000h to FFFFh. Figure 4-3 shows the memory mapping of the P89C51RD2BN microcontroller.

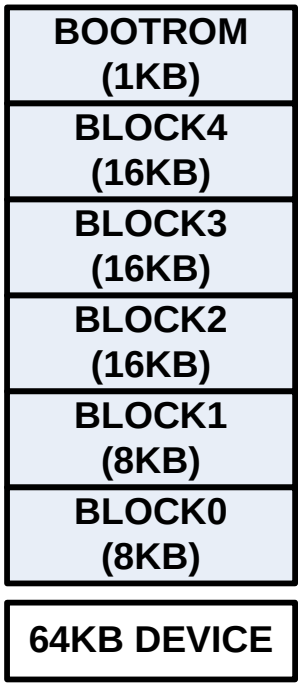


Figure 4-3: Memory Map of P89C51RD2BN.

The on-chip memory is divided into blocks of 8KB, 16KB and 1KB for Boot ROM. The Boot ROM is shadowed with the user code memory in the address range from FC00h to FFFFh. This Boot ROM contains a set of instructions which allows the microcontroller to perform a number of flash programming and erasing functions. The Boot ROM also provides the communications through the serial port. The use of Boot ROM is the key to the concept of design of SU51 MB. When the DIP SW-1 is “ON” the execution starts at FC00h, which executes the on-chip Boot ROM contents and allows the hex file to be downloaded. For the opposite case when the SW-1 is “OFF” the execution starts at 0000h. So the user codes can be executed.

Ports and Connectors

All four ports of the microcontroller are available on connectors for different interfacing options.
The connector details are as follows:

CONNECTOR	**DETAILS**
P0	PORT0
P1	PORT1
P2	PORT2
P3	PORT3
J1	SERIAL DB-9F
J2	POWER CONNECTOR
SU1	POWER CONNECTOR
SU2	POWER CONNECTOR

Connection Details and Signal Definitions

J2 +5V power connector.
Upper : +5V
Lower : GND

SU1 and SU2 +5V power connector.
Upper : GND
Lower : +5V

J1 Serial DB-9 Female Connector
1 : GND
2 : PIN-7 MAX-232
3 : PIN-8 MAX-232
4 : NC
5 : GND
6 : NC
7 : NC
8 : NC
9 : NC

P0 PORT0 CONNECTOR
1 : P0.0
2 : P0.1
3 : P0.2
4 : P0.3
5 : P0.4
6 : P0.5
7 : P0.6
8 : P0.7

P1 PORT1 CONNECTOR
1 : P1.0
2 : P1.1
3 : P1.2
4 : P1.3
5 : P1.4
6 : P1.5
7 : P1.6
8 : P1.7

P2 PORT2 CONNECTOR
1 : P2.0
2 : P2.1
3 : P2.2
4 : P2.3
5 : P2.4
6 : P2.5
7 : P2.6
8 : P2.7

P3 PORT3 CONNECTOR
1 : P3.0
2 : P3.1
3 : P3.2
4 : P3.3
5 : P3.4
6 : P3.5
7 : P3.6
8 : P3.7

Configuration and installation

The SU51 MB requires +5V power and a serial connector DB-9 to connect with PC running WINISP.

Hardware and Software requirements for SU51 MB:

- The SU51 MB.
- A serial cable, i.e. 9-Pin male (from board) to 9-pin female (to PC).

- A PC with an available RS-232 Port, 128 MB RAM and 2.5 MB Hard Disk space.
- Microsoft Windows 98/ME/NT/XP/2000
- WINIPS software.

Configuration

The board can be configured in two modes depending on the user's selection of DIP switch setting. The switch-1 of the DIP Switch is dedicated for enabling the appropriate mode selection i.e. Program or Execution. The details of the on board DIP-Switch is given in Figure 4-4 as shown below.

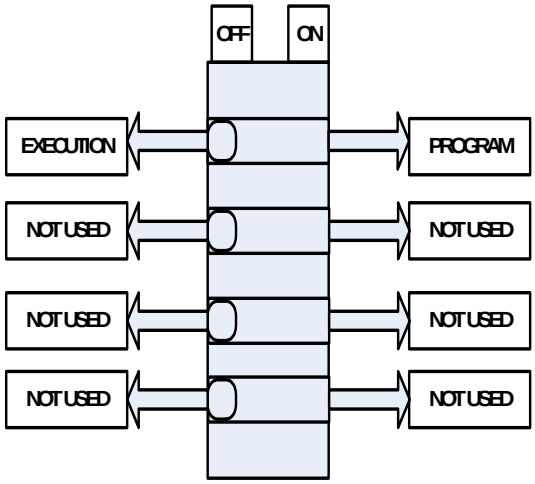


Figure 4-4: DIP SW setting for program and execution mode.

Installation

- Connect +5V power supply to the board.
- Connect Serial cable, 9-Pin Female connector to PC and 9-Pin Male connector to SU51 MB.
- Depending on mode selection or DIP SW-1 settings the user can perform the task.
- Install WINISP into the PC by running the WinISP.exe provided with SU51 MB.
- WINISP, a software utility to implement ISP programming with a PC, is available from the Philips website (www.semiconductors.philips.com).
  The Program mode is set by keeping the DIP SW-1 "ON" then using WINISP the on board chip can be easily programmed.

How to Use WINISP

The user has to build the *.asm file in editor or notepad and assemble the file using any assembler to create the *.hex file or any C-Compiler in case the codes are written in C to get the *.hex file. This hex file is easily downloaded into the SU51 MB using WINISP.
The following steps explain the user about the WINISP environment.

1. Start the WINISP software. The following start-up screen will appear as shown in the Figure 4-5.

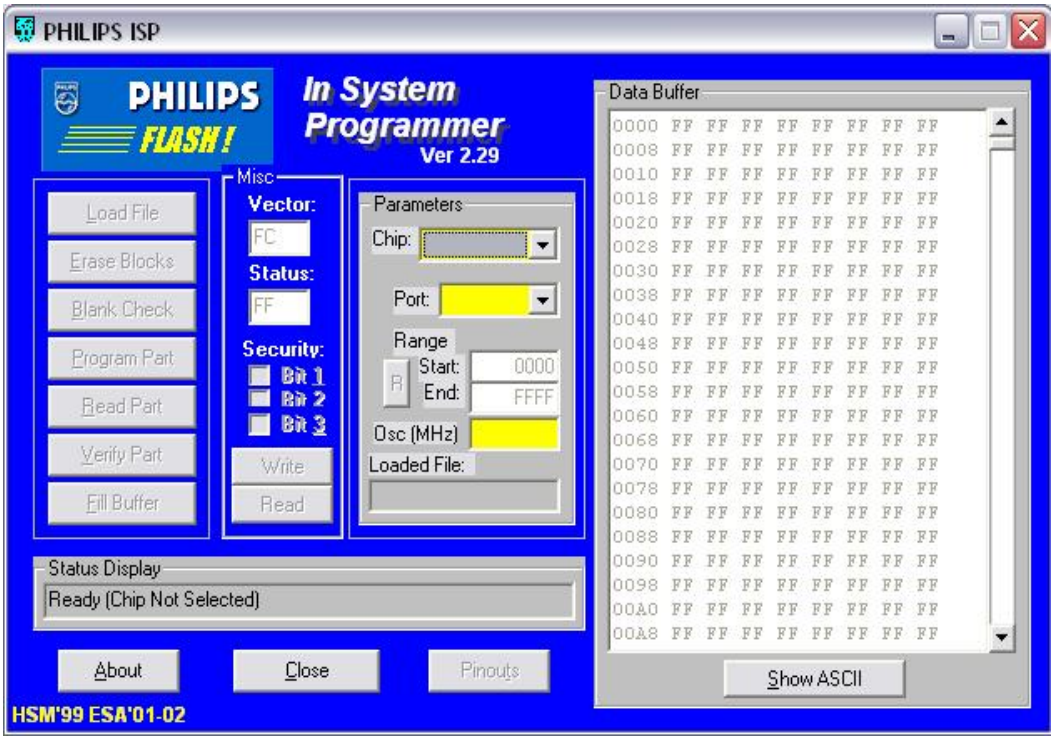


Figure 4-5: Start-up screen for WINISP.

2. Select the Chip: P89C51RD2, Port: COM1 (available port to which SU51 MB serial cable is connected), Osc (MHz): 11.0592. Pictorial representation of that is given in Figure 4-6 below:

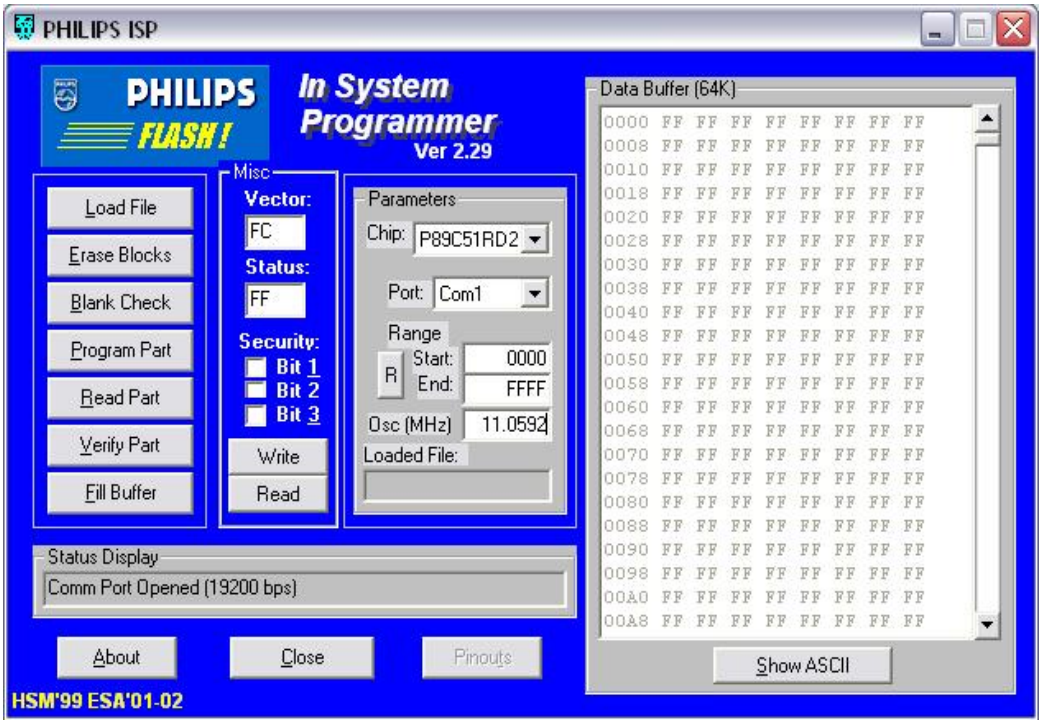


Figure 4-6: WINISP Set-up for SU51 MB.

3. To program the microcontroller user has to load *.hex file first. This can be done by clicking in the “Load File” button which will allow to select the *.hex file from the path. For example AAB.hex file is selected. The Figure 4-7 below shows the loaded file.

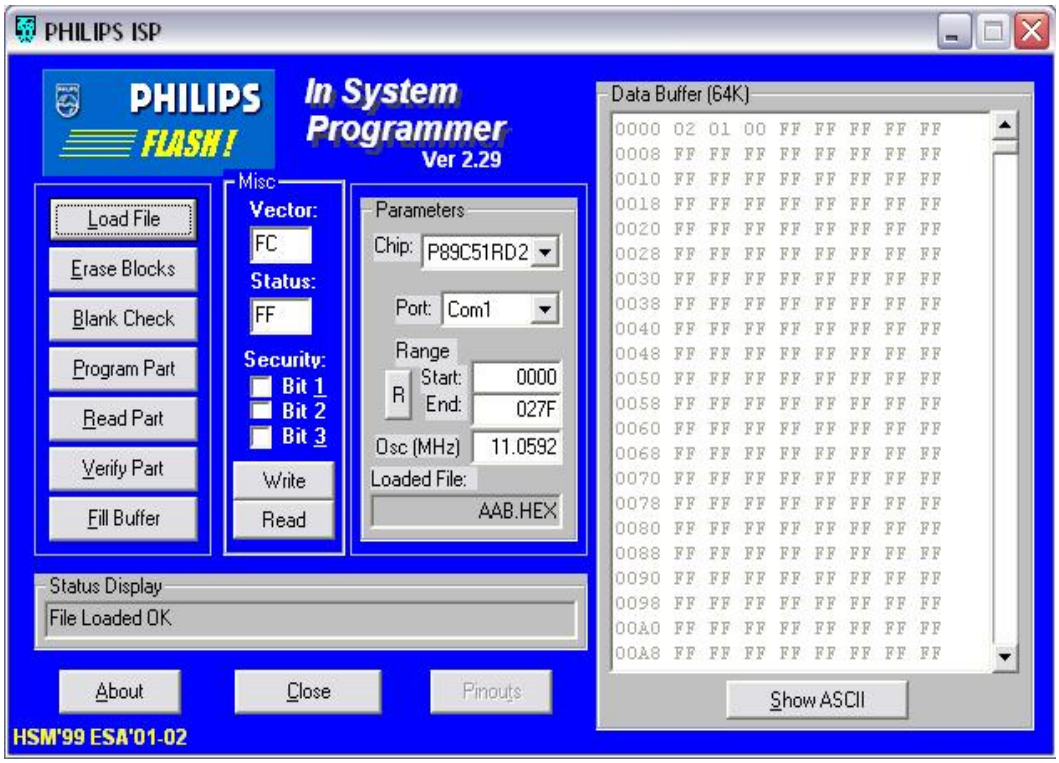


Figure 4-7: WINISP with AAB.hex file loaded.

4. While using the board first time Set the Vector: FC and Status: 0 by simply typing the values and click the “write” button below. Do not select any Security bit. Now click on the “read” button. This will display the set values of vector and status, if the other than above set value appears then do the same thing again. One can see the following changes in the WINISP as shown in Figure 4-8 below.

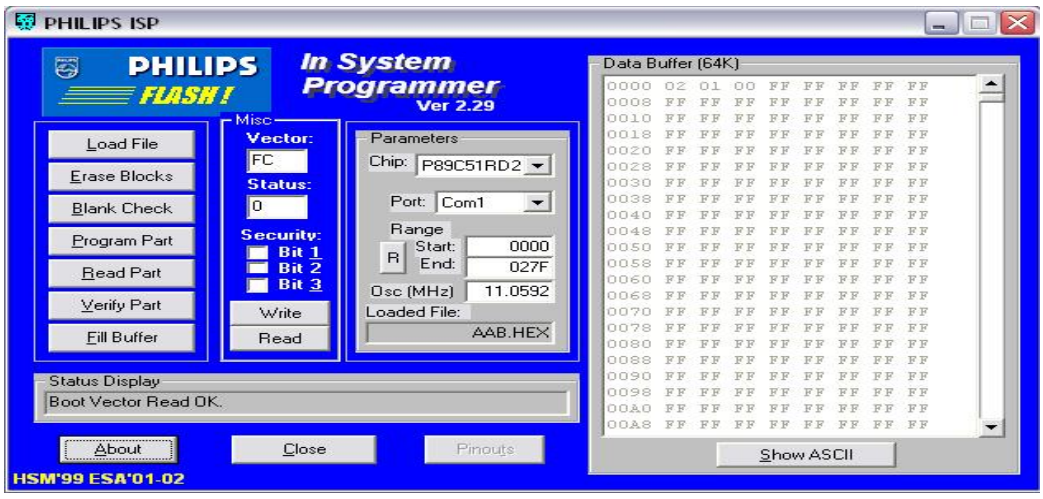


Figure 4-8: WINISP Boot Vector: FC & Status: 0.

5. Now depending on the *.hex file size click on the “Erase Blocks” button this will provide user to erase any on-chip block as shown in Figure 4-9 below.

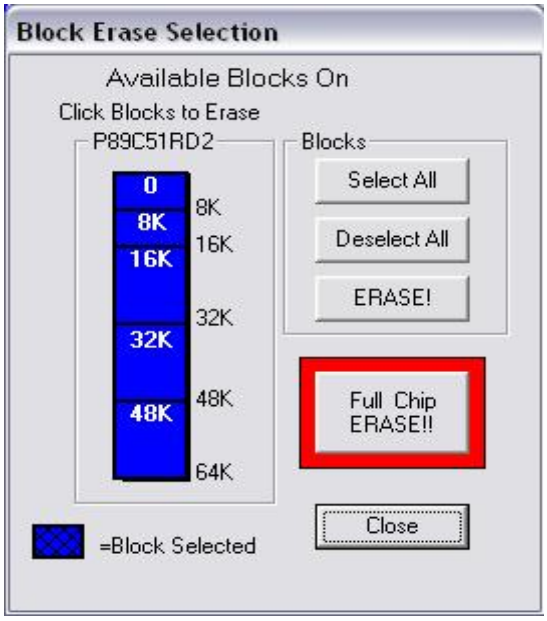


Figure 4-9: WINISP Erase Block.

It is recommended not to perform “Full Chip ERASE!!” so after selecting the appropriate block depending on the size of *.hex file click on “ERASE!” button. Then click on “Close” button to again go to the previous menu.

6. After erasing click on “Program Part” button to download the *.hex file in to the flash of microcontroller. The program execution will start from 0000h.

After programming the chip switch off the power of SU51 MB. Now select the “Execution” by keeping the DIP SW-1 “OFF”. Power up the board this will execute the user downloaded *.hex (AAB.HEX) file. If problem persists then press the RESET button which will force the program to again restart.

Component layout diagram

Figure 4-10 shows the top-side component layout diagram for the SU51MB.

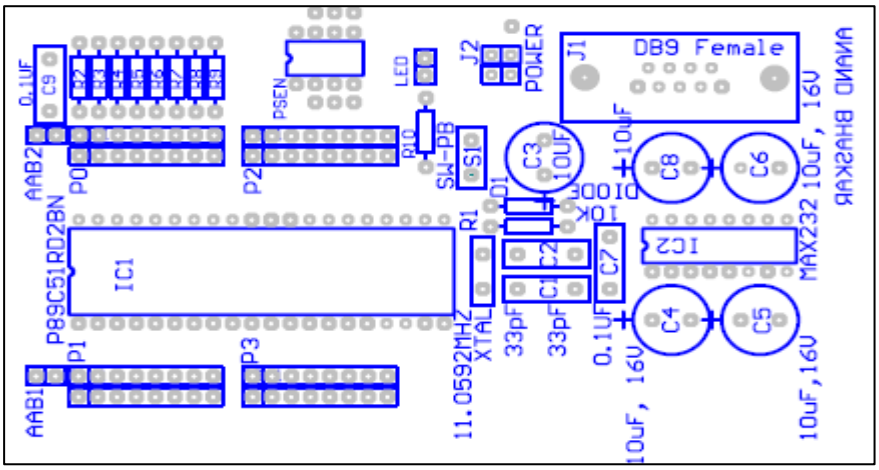


Figure 4-10: Component layout diagram.

Part List for the SU51MB

The components used in the design of SU51MB are tabulated below in Table 4-1.

Part	Used	Part Type	Designators
1	4	Connector	PORT0 PORT1 PORT2 PORT3
2	2	0.1UF	C7 C10
3	1	1.2K	R2
4	1	1N914	D1
5	8	3.3K	R3 R4 R5 R6 R7 R8 R9 R10
6	1	10K	R1
7	2	10UF	C3 C8
8	1	10uF, 16V	C4
9	1	10uF, 16V	C5
10	1	10uF, 16V	C6
11	1	11.0592MHZ	XTAL
12	2	33pF	C1 C2
13	1	DB9 Female	J1
14	1	LED	RED
15	1	MAX232	U2
16	1	P89C51RD2BN	U1
17	1	SW-PB	S1
18	1	SW SPDT	S3

Table 4-1: Part-list for SU51MB.

Solder Side PCB for SU51MB

The solder side PCB for the SU51MB is shown in the Figure 4-11 below.

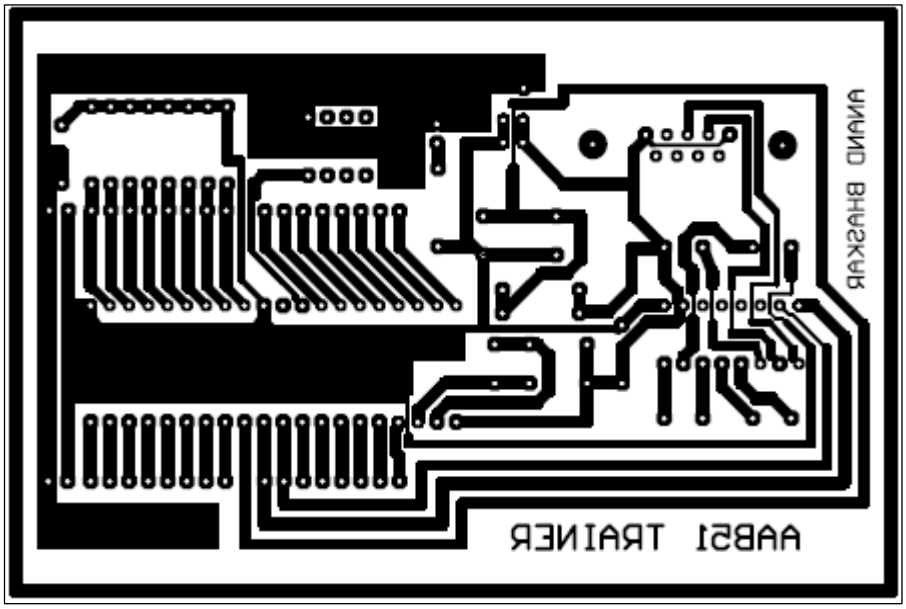


Figure 4-11: Solder side (bottom) PCB for the SU51MB.

4.6 Using the SU51MB

The step by step procedure to use the SU51 MB for any embedded system design and development is presented. Initially user has to build an assembly or C language code file with its appropriate extension depending on the assembler used. Then by means of any available assembler or compiler user has to build the hex file. That must be of Intel Hex format. Following example shows the test program for the kit.

Test Program for SU51MB

To test the SU51MB hardware connection, user can run a simple test in which all the bits of P1 toggle continuously with some delay in between the “on” and “off” states. The program for the SU51 MB in assembly is provided below.

```
                MOV   89H, #10H    ; Timer 1, mode 1 (16-bit)
WHOLE:          MOV   8BH, #00H    ; TL1=00h, low byte
                MOV   8DH, #00H    ; TH1=00h, high byte
                MOV   A, #0F0H
                MOV   90H, A       ; Data 0F0h out on Port1
                ACALL DELAY        ; Delay
                CPL   A            ; Complement Acc. contents
                MOV   90H, A       ; Output on Port1
                ACALL DELAY        ; Delay
                SJMP  WHOLE        ; Continuous loop
DELAY:          MOV   R5, #0FFH    ; delay subroutine
                MOV   88H, #60H    ; Set TF
PON:            JNB   8FH, PON     ; Stay until timer rolls over
AGAIN1:         MOV   R4, #0FFH    ; Adjust for 1 second delay
AGAIN:          NOP
                DJNZ  R4, AGAIN
                DJNZ  R5, AGAIN1
                MOV   88H, #00H    ; Clear TF for next round
                RET
                END
```

User has to type the program in DOS editor or Notepad and save it as IO.asm for the above test program. Let us see how to assemble the IO.asm file. Follow the following simple steps to assemble the IO.asm file saved at path c:\IO.asm:

1. Open the ASM51 by double clicking the ASM.exe. This will open the below Pop-up screen Figure 4-12.

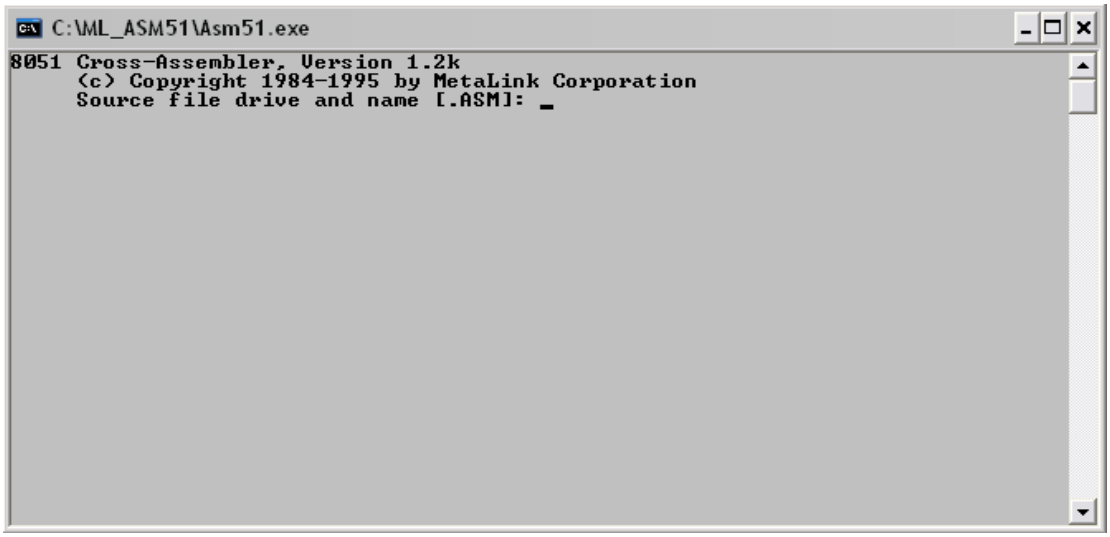


Figure 4-12: ASM51 start-up screen.

2. Write the source file drive and name [.ASM]: C:\IO.ASM and press “Enter”.
3. This will create the IO.hex file on the same path where the .asm file is saved.
4. This IO.hex file is now downloaded into the microcontroller by using the WINISP software. Go through the section 2.3 HOW TO USE WINISP page no.6. Before using WINISP user has to select the program mode by keeping the DIP SW-1 “ON”. After programming the chip user has to set the Status = 00h, to start the execution of program from 0000h. In program mode the controller starts it execution at FC00h where the boot vector is loaded to execute the boot loader.
5. After programming the controller, user has to set the DIP SW-1 to execution mode i.e. “OFF”.
6. Restarting the SU51 MB will now execute the user program that is downloaded in the flash by WINISP.
7. Connect 8-LED’s to the Port1, and observe the Port1 nibble will toggle with the delay defined in the program.

The connection diagram for the LED’s to the microcontroller is shown in Figure 4-13.

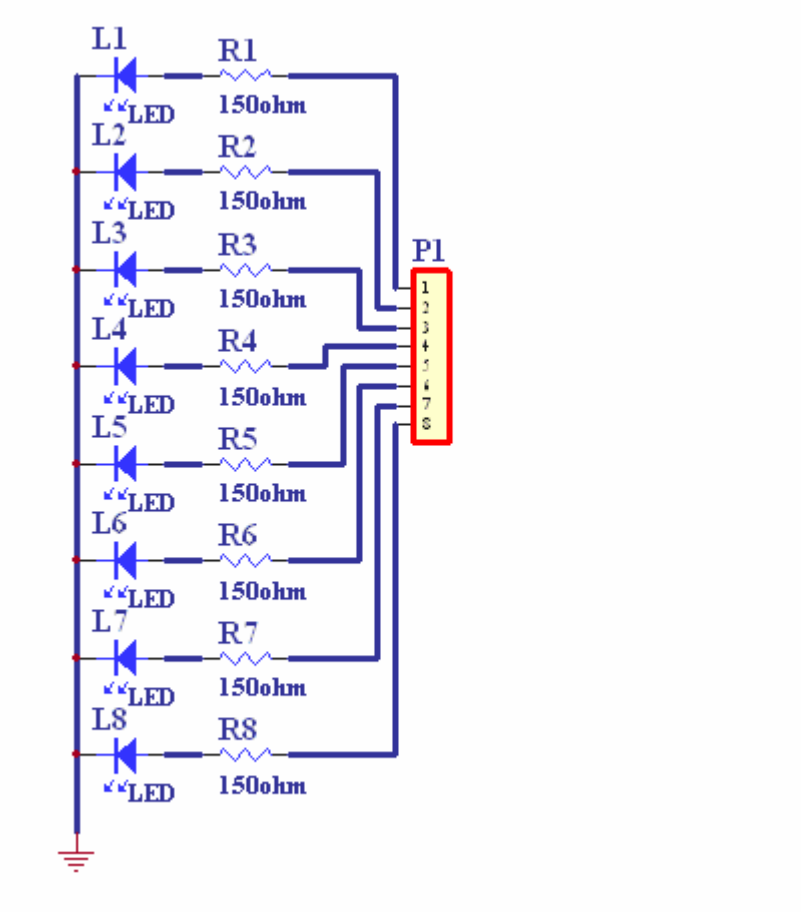


Figure 4-13: LED's connection to Port1 of SU51MB.

The whole setup of the SU51MB in the research laboratory is shown in the Figure 4-14.

Figure 4-14: SU51MB setup in the embedded system laboratory.

4.7 Troubleshooting Tips

Running the test program on SU51 MB should toggle the Port1 nibbles with some delay. If the system does not work, follow these steps to find the problem.

- With the power off check all the connections for all pins, especially Vcc and Gnd.
- Check RST (Pin No. 9) using an oscilloscope. When the system is powered up, Pin No. 9 is low. Upon pressing the push-button switch it goes high.
- Observe the XTAL2 pin on the oscilloscope while power is on. You should see a crude square wave. This indicates that the crystal oscillator is good.
- If still the problem persists then check the communication of PC and SU51 MB using hyper-terminal program. Do all the connections and follow steps to establish communication:
    1. Power up the trainer after all connections, set the switch to program position before power up.
    2. In Windows Accessories, click on Hyper Terminal. If a model installation option appears choose “NO”.
    3. Type a name, and click OK.
    4. For “Connect Using” select COM1 and click OK. Choose other serial port if COM1 is busy.
    5. Pick 9600 baud rate, 8-bit data, no parity bit, and 1 stop bit.
    6. Change the “Flow Control” to NONE or Xon/Xoff, and click OK.
    7. This will open the start up screen for windows hyper terminal. In that write “U” it will echo the same character “U” if the SU51 MB is working properly.

Figure 4-15 shows the screen shot of hyper terminal with working SU51 MB.

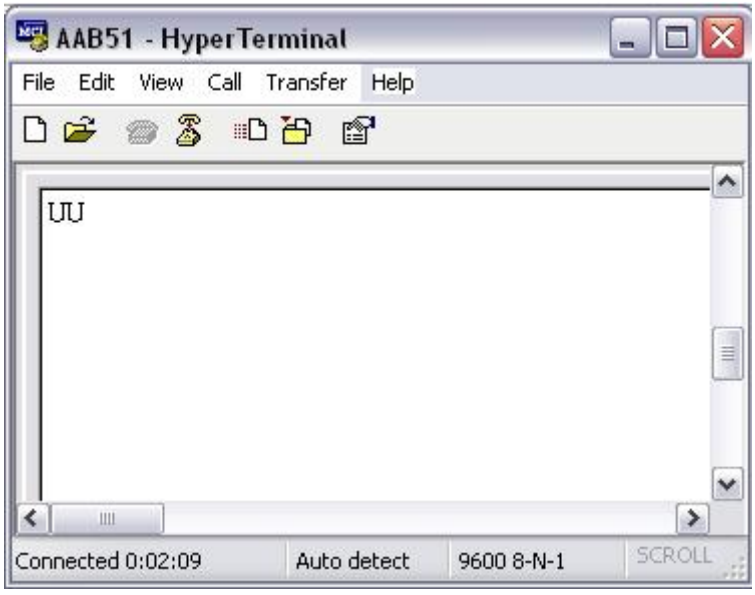


Figure 4-15: Checking the working of SU51 MB using Hyper Terminal.

Result

Port1 connections are perfect as the connected LEDs are toggling. Experimental board SU51 MB is tested and ready to use for further advanced experiments.

4.8 Future Developments

Various interface cards can be developed by using SU51 MB as development platform and can be applied to various applications. To provide the universal platform for microcontroller based system design and to provide all the facilities on the single platform for example the universal robo-panel boards are now available in the markets. Also the design cycle is inadequate without a debugging tool. Debug facility can be provided by using the PC based debugger as presented in the next chapter.

Chapter 5 Debugging Tool for SU51MB: PICE

One of the development cycles for the embedded system design is emphasized in this chapter. Programmable In-Circuit Emulator (PICE) based techniques and experiment is developed. PICE is used as an external debugger, which supports user target microcontroller. An experimental setup that provides the universal debugging tool for MCS51 family of microcontrollers is presented in this work to learn the usefulness of the PICE as a debugging tool. The drift between the real time clock and system clock is calculated by using the PICE.

5.1 Introduction

In the current scenario embedded microcontroller development cycle employ many different techniques and development tools. People involving in embedded system design deals with many flavors of microcontrollers and use assembly & higher level languages to program them. In the development phase one can use the Programmable In-Circuit Emulator (PICE) from the beginning i.e. all development in the IDE of PICE or using it as external debugger as done in the present work. Suppose designer is using external development tool, which supports the target microcontroller and developed the loadable module. Once the module is ready, the PICE emulator can be used to load the program module for debugging. The program module in present work is in terms of io.hex files generated by assembling the simple MCS51 program (io.asm) using ASM51 assembler. Once the program module is loaded on the PICE software, PICE emulator software environment is used for debugging the software.

5.2 PICE-52 Hardware & Software Installation

Figure 5-1 shows the emulator hardware with its connectors and indicators.

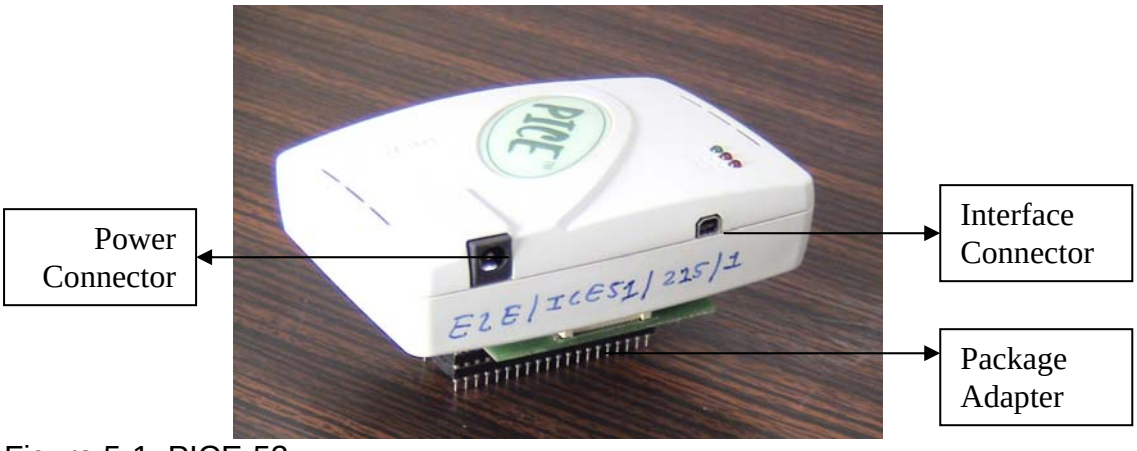


Figure 5-1: PICE-52

The PICE hardware sits on the target microcontroller board and emulates the target microcontroller and checks for its proper working. Package adapter 40-pin DIP is to be inserted in the target board [SU51MB]. PICE-52 software

[Provided by Phyton] must be installed in the PC before one uses the PICE hardware. The block diagram of the practical setup is shown in Figure 5-2.

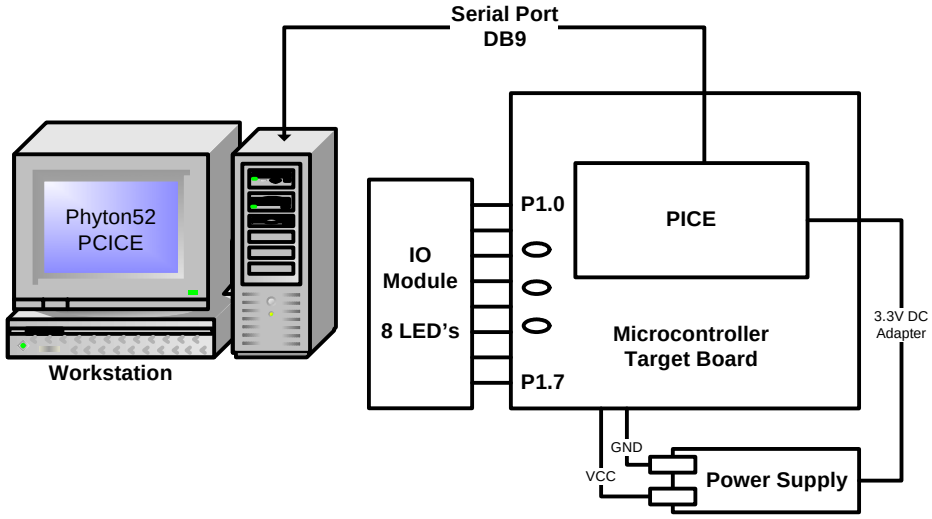


Figure 5-2: Block Diagram of practical setup.

5.3 Preparing the PICE

- Put the PICE carefully on the target microcontroller board. Connect the serial connector between the PC and the PICE. Provide proper biasing (+5V DC) to the PICE and microcontroller board from power adapter.
- Open the PICE-52 In-Circuit Emulator from the start menu by single clicking on it. This will allow you to configure the communication between the PC and PICE. PICE-52 configuration dialog box will open as shown in Figure 5-3.

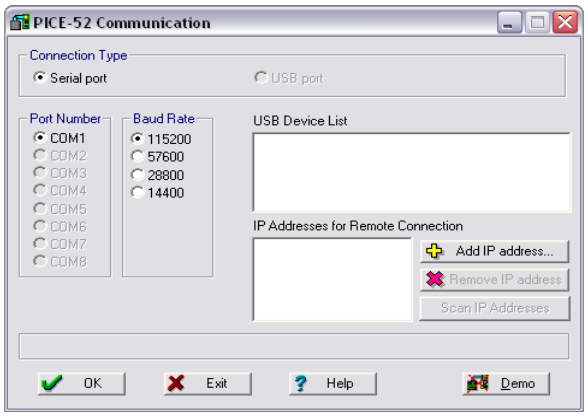


Figure 5-3: PICE-52 First Window.

Do the setting as shown in the Figure 5-3 for proper data transfer between PC and PICE. Press OK to start the PICE-52 In-Circuit Emulator. The startup test will be performed by the PICE-52 automatically as shown in Figure 5-4.

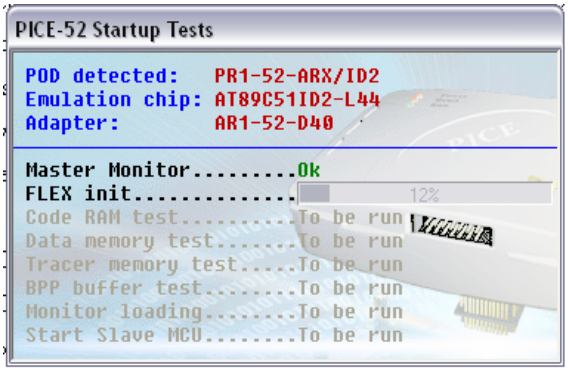


Figure 5-4: PICE-52 Startup tests Window.

If the PICE & SU51 MB is not biased then the PICE-52 opens in the demo mode, otherwise the examples pop-up window will be open as shown in Figure 5-5.

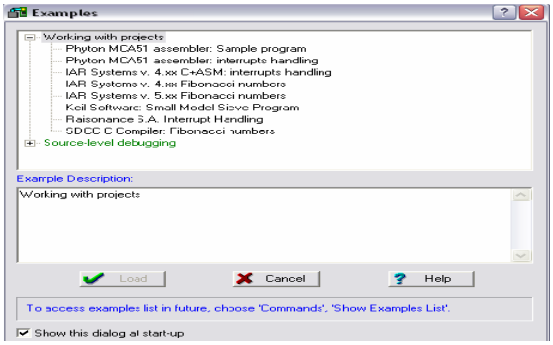


Figure 5-5: PICE-52 Startup Examples Window.

Click cancel in the example window and close to start the PICE-52 In-Circuit Emulator. Figure 5-6 shows the first screen of the PICE-52 software.

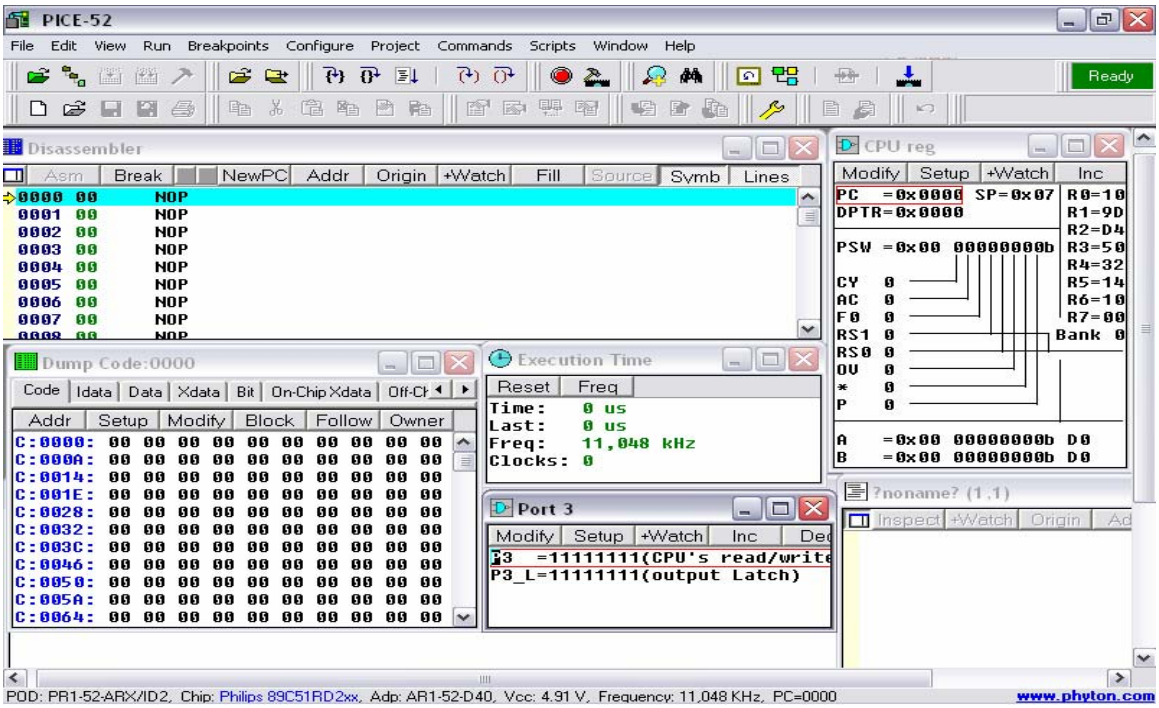


Figure 5-6: PICE-52 First Screen.

The IDE shown in Figure 5-6 is used to debug the target microcontroller based embedded design and can be configured according to the user requirement. As in this case only five windows are in use to debug the small programming example.

5.4 PICE as Debugging Tool

Once the PICE-52 hardware as well as software is prepared, user has to load the program module developed for target microcontroller system. Here we used the PICE as external debugging tool and io.asm file is assembled and io.hex file is generated as discussed in the previous chapter. As shown in Figure 5-6 click on the File menu and Load program for debugging. A pop-up dialog box will appear, browse the io.hex file and in compiler vendor box click on the Standard/Extended Intel Hex and press ok. This will load the program module to debug. To run the program loaded use the following commands:

F7	:	Single Step
F9	:	Run/Stop
F5	:	MCU Reset
Ctrl+F5	:	Time Counter Reset

By using the single stepping one can easily debug the program module for its correctness for attached hardware. One can also run the whole source module and check the working of the hardware for its correctness.

5.5 Using the PICE

Two activities are presented here to understand the proper working of the PICE.

A. To debug the IO ports

Test the operation of the port1 of SU51 MCU as follows. Assemble and load the io.hex file into the PICE IDE as discussed above. The test program toggles the port1 of the 8051 and connected LED's. To watch the bits of the ports toggle on and off user has to use the single step mode. Program module is given below:

```
;IO MODULE PROGRAM to toggle port1
;Mainline
            ORG  20H
      P1    EQU  90H          ;Equate port1 addrs
AGAIN:      MOV  A,#55h
            MOV  P1,A
            CPL  A
            MOV  P1,A
            JMP  AGAIN
            END               ;End of prog
```

Figure 5-7 shows the output on the IO module for the program running in the PICE IDE.

Figure 5-7: Result of IO.ASM - Executing

B. To Calculate the Time Delay

PICE also provides Real-time Timer and Frequency Measuring System. The heart of the microcontroller is the crystal providing the regular clock pulses to it depending on its working frequency. In real time application calculation of time to execute the instruction and time delay is very critical. This can be done manually by calculating individual instruction time to execute and add them all to get the total time delay. This will lead to inaccuracy as drift between the real time and the time provided by the crystal clock. As a crystal of 11.0592 MHz will not give the exact frequency all the time, it may fluctuate above or below to the provided value.

Let us observe the drift between the real time clock and clock provided by the crystal attached to the microcontroller system. Firstly find the time of delay for the following subroutine manually, assuming a crystal frequency of 11.0592 MHz.

		MACHINE CYCLE
Delay:	MOV R3, #10	1
HERE:	NOP	1
	NOP	1
	NOP	1
	NOP	1
	DJNZ R3, HERE	2
	RET	1

The time delay inside the HERE loop is [10 (1+1+1+1+2)] x 1.085 µs = 65.1 µs. Adding the two instructions outside the loop we have 65.1µs+2x1.085µs = 67.27 µs.

The same thing can be easily calculated by PICE and shown in the Execution time window. Load the above program module in the PICE IDE and do the single step, every time it will measure the accurate system clock and give the time to execute the single instruction. In our case it is giving 68.43 µs time to execute the above program module as shown in the figure 5-8.

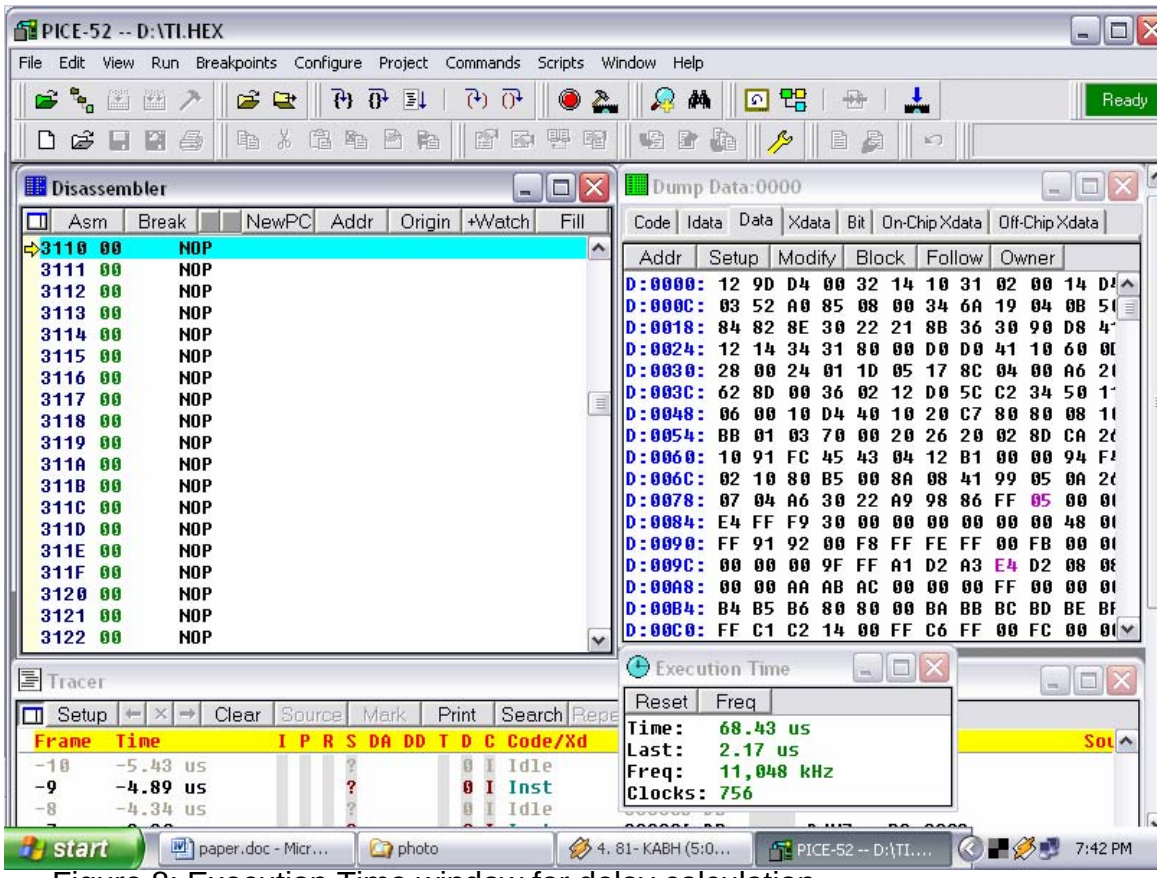


Figure 8: Execution Time window for delay calculation.

The drift between the real and system clock can be calculated as 68.43 - 67.27 µs =1.16 µs. This can be quite critical for Real Time Embedded Systems.

RESULTS

A) PCICE cab be used as an external debugger for the testing of proper working of the target system by checking the port1 connected LED's toggle by writing a simple program io.asm.
B) The drift between the real time clock and the system clock is calculated.

CONCLUSION

Only two perspective of the PCICE are covered in the present work. First as an external debugger and secondly to provide accuracy to the time constraint applications. It can be used as real time simulator for any development system.

Chapter 6 ARM7: LPC-2129 Board

This chapter provides the details of the ARM7 architecture based LPC-2129 prototype board which can be used in any relevant application. In this work it is utilized as the prototype board for the ARM processor based application development and embedded system design.

6.1 Introduction

The LPC2129 is based on a 16/32 bit ARM7TDMI-S™ CPU with real-time emulation and embedded trace support, together with 128/256 kilobytes of embedded high speed flash memory. A 128-bit wide memory interface and unique accelerator architecture enable 32-bit code execution at maximum clock rate. For critical code size applications, the alternative 16-bit Thumb Mode reduces code by more than 30% with minimal performance penalty.

With their compact 64 pin package, low power consumption, various 32-bit timers, 4-channel 10-bit ADC, 2 advanced CAN channels, PWM channels and 46 GPIO lines with up to 9 external interrupt pins these microcontrollers are particularly suitable for automotive and industrial control applications as well as medical systems and fault-tolerant maintenance buses. With a wide range of additional serial communications interfaces, they are also suited for communication gateways and protocol converters as well as many other general-purpose applications.

The main features of the board are listed below.

- LPC 2129/2119 Single-chip 16/32-bit microcontrollers; 128/256 KB ISP/IAP Flash with 10-bit ADC and CAN
- A 7-Segment LED Display
- 2 CAN Ports
- A Serial Port for ISP
- An RS232 Serial Port
- A Box Header for JTAG Wiggler connection
- DC Power Supply Connector
- Optional USB Socket for Power Supply
- Works on +5 to +7.5V Power Supply
- ARM Core works on +1.8V Power
- Peripherals work on +3.3V Power
- Other On-Board Chips work at +5V Power

Software development tools for the current work are as listed below.

- Keil uV3 or WINARM for Programming
- H-JTAG or Phillips Flash Utility for Flash Programming.
- MS Windows 98/ME or windows WT/2000/2003/Xp

6.2 Block Diagram and Schematic

The block diagram for the LPC-2129 board is shown in the Figure 6-1. The block diagram consists of the LCP-2129 chip and its support circuitry which will combine and provide features to the prototype board as listed above.

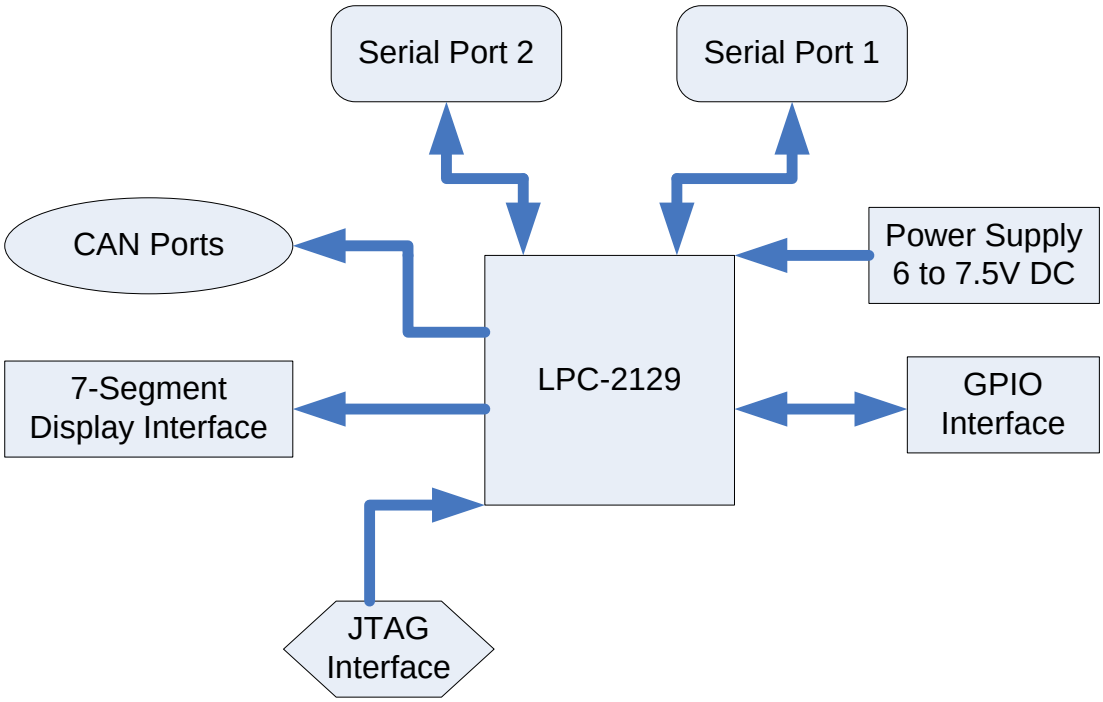


Figure 6-1: Block Diagram ARM7 LCP-2129 Board.

The schematic diagram for the board is shown in Figure 6-2. The board is designed around ARM7 LPC2129 microcontroller. LPC2129 is basically a 32-bit device which can provide code efficiency by thumb instruction set support and behave as 16-bit microcontrollers. For this device two GPIO, two serial ports, two CAN ports and ADC is provided on-chip.

The board circuitry consists of eight LED's connected to the Port1 (P1.16 to P1.23) to check the working of the GPIO of the LPC-2129 microcontroller. All the GPIO are taken out separately around the LPC-2129 chip for the further experimentation. The board is running at 12MHz clock. Proper power to the controller is provided by using 5.0v, 3.3v and 1.8v voltage regulators. Two serial ports (Port1 and Port2) DB9 are provided in the design of the board. Port2 is used to program the microcontroller flash when the switch SW1 is set on the PRG (program) side. To run the user program after dumping the codes into the flash, user mode must be activated by putting SW1 on the other side. The microcontroller also can be programmed by the JTAG interface. So both serial and JTAG programming of the microcontroller memory is supported by this prototype type and can sit easily in any embedded development design or application.

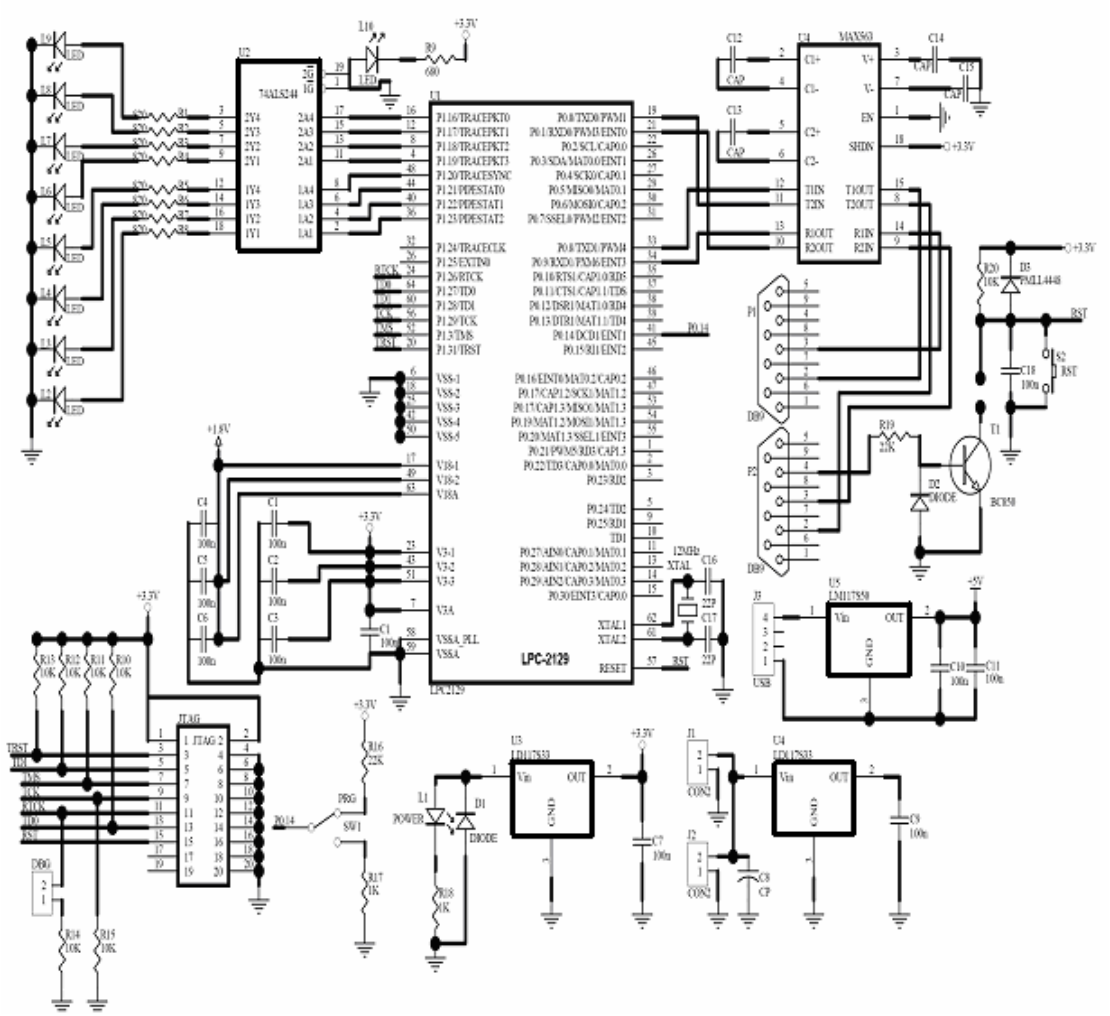


Figure 6-2: Schematic Diagram of the LPC-2129 Board.

6.3 Hardware Details

The development board is mounted around the ARM7 LCP-2129 microcontroller. The pin configuration for the LPC-2129 microcontroller is shown in the Figure 6-3.

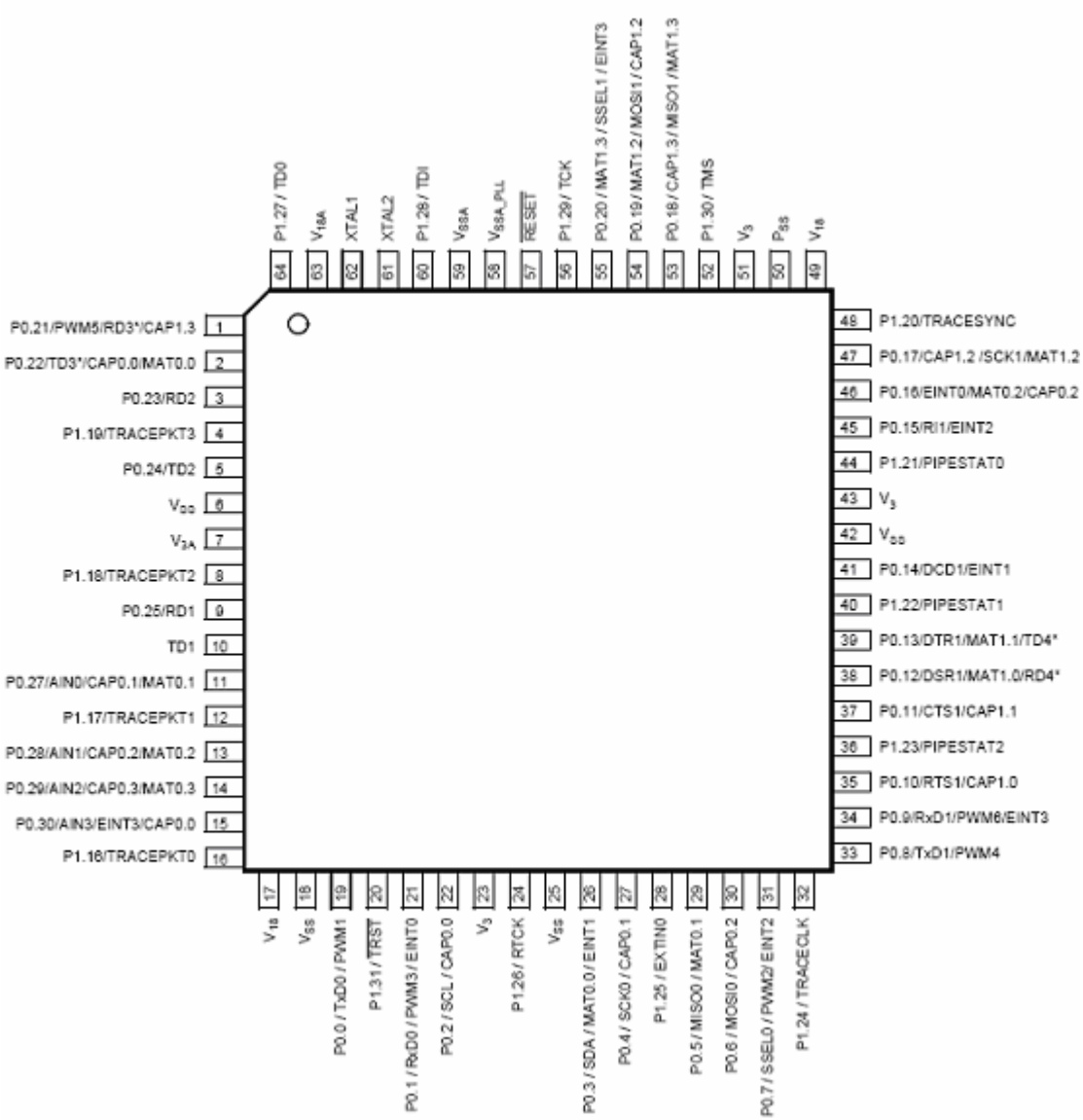


Figure 6-3: LPC-2129 64-pin package.

Features

- 16/32-bit ARM7TDMI-S microcontroller in a 64 pin package.
- 16 kB on-chip Static RAM
- 128/256 kB on-chip Flash Program Memory (at least 10,000 erase/write cycles over the whole temperature range). 128-bit wide interface/accelerator enables high speed 60 MHz operation.
- In-System Programming (ISP) and In-Application Programming (IAP) via on-chip boot-loader software. Flash programming takes 1 ms per 512 byte line. Single sector or full chip erase takes 400 ms.
- Embedded ICE-RT interface enables breakpoints and watch points. Interrupt service routines can continue to execute whilst the foreground task is debugged with the on-chip RealMonitor software.
- Embedded Trace Macrocell enables non-intrusive high speed real-time tracing of instruction execution.
- Two/four interconnected CAN interfaces with advanced acceptance filters.

- Four/eight channel (64/144 pin package) 10-bit A/D converter with conversion time as low as 2.44 ms.
- Two 32-bit timers (with 4 capture and 4 compare channels), PWM unit (6 outputs), Real Time Clock and Watchdog.
- Multiple serial interfaces including two UARTs (16C550), Fast I2C (400 kbits/s) and two SPIs™.
- 60 MHz maximum CPU clock available from programmable on-chip Phase-Locked Loop.
- Vectored Interrupt Controller with configurable priorities and vector addresses.
- Up to forty-six (in 64 pin package) and hundred-twelve (in 144 pin package) 5 V tolerant general purpose I/O pins. Up to 12 independent external interrupt pins available (EIN and CAP functions).
- On-chip crystal oscillator with an operating range of 1 MHz to 30 MHz.
- Two low power modes Idle and Power-down.
- Processor wake-up from Power-down mode via external interrupt.
- Individual enable/disable of peripheral functions for power optimization.
- Dual power supply.
  - CPU operating voltage range of 1.65V to 1.95V (1.8V +/- 8.3%).
  - I/O power supply range of 3.0V to 3.6V (3.3V +/- 10%).

The features of the LPC-2129 which are utilized further in this work are discussed below.

GPIO

The general purpose input/output provides direction control of individual bits, separate control of output set and clear. After RESET all I/O devices set default to inputs. This can be used as general purpose I/O, for driving LED's or other indicators, controlling off-chip devices, and sensing digital inputs. Table 6-1 describes the pin description of GPIO.

Pin Name	Type	Description
P0.0 – P0.31 P1.16 – P1.31	Input/ Output	General purpose input/output.
P2.0 – P2.31 P3.0 – P3.31	Input/ Output	External bus data/address lines shared with GPIO, digital and analog functions. The number of GPIOs/digital and analog functions actually available depends on the selected bus structure.

Table 6-1: GPIO pin description.

LPC-2129 has two 32-bit General Purpose I/O ports. Total of 30 out of 32 pins are available on PORT0. PORT1 has up to 16 pins available for GPIO functions. PORT0 and PORT1 are controlled via two groups of 4 registers as shown in Table 6-2.

Name	Description	Access
IOPIN	GPIO Port Pin value register. The current state of the GPIO configured port pins can always be read from this register, regardless of pin direction and mode. Activity on non-GPIO configured pins will not be reflected in this register.	Read Only
IOSET	GPIO Port Output set register. This register controls the state of output pins in conjunction with the IOCLR register. Writing ones produces highs at the corresponding port pins. Writing zeroes has no effect.	Read/Write
IODIR	GPIO Port Direction control register. This register individually controls the direction of each port pin.	Read/Write
IOCLR	GPIO Port Output clear register. This register controls the state of output pins. Writing ones produces lows at the corresponding port pins and clears the corresponding bits in the IOSET register. Writing zeroes has no effect.	Write Only

Table 6-2: GPIO register description.

UART0 and UART1

Both UART0 and UART1 are identical with each other with a small addition in UART1 of a modem interface. These are the 16 byte receive and transmit FIFOs. They both are having built-in baud rate generator. UART0 and UART1 pin description is given in Table 6-3. The resistor description is given in Table 6-4.

Pin Name	Type	Description
RXD0	Input	**Serial Input**. Serial receive data.
TXD0	Output	**Serial Output**. Serial Transmit data.
RXD1	Input	**Serial Input**. Serial receive data.
TXD0	Output	**Serial Output**. Serial transmit data.

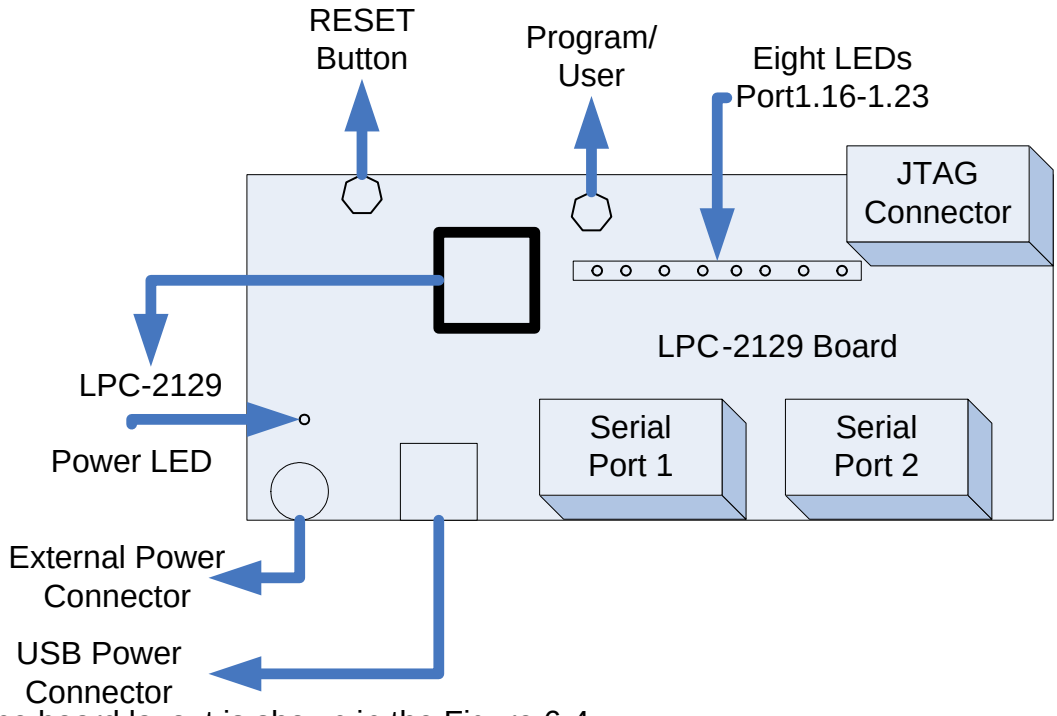
Table 6-3: UART0 and UART1 pin description.

Name	Description	Access
U0RBR/U1RBR	Receive buffer register	RO
U0THR/U1THR	Transmit holding register	WO
U0IER/U1IER	Interrupt enable register	R/W
U0IIR/U1IIR	Interrupt ID register	RO
U0FCR/U1FCR	FIFO control register	WO
U0LCR/U1LCR	Line control register	R/W
U0LSR/U1LSR	Line status register	RO
U0SCR/U1SCR	Scratch pad register	R/W
U0DLL/U1DLL	Divisor latch LSB	R/W
U0DLM/U1DLM	Divisor latch MSB	R/W

Table 6-4: UART0 and UART1 register description.

Getting Started

- Connect the Power Supply using a DC adapter of 6 to 7.5 volts to the DC Socket.
- Or connect USB Cable from PC to USB socket of the ARM7 LPC-2129 board to provide Power Supply.
- Once the board is powered up the Power ON LED should Glow.
- All the eight LEDs connected with the port1 will glow in the user mode and in the program mode the eight LED will not glow.



The board layout is shown in the Figure 6-4.

Figure 6-4: LPC-2129 Board layout diagram.

LPC2129 Board Description

1. Peripherals

Unit	**Description**
COM Port 1	RS232 DB9 Female connector for LPC2129 UART0.
COM Port 2	RS232 DB9 Female connector for LPC2129 UART1.
JTAG Connector	2x10 0,1" step connector for programming with ARM-JTAG.
Buttons	Reset and Program or User mode select button.
LEDs	Power LED and Eight GPIO LEDs.

2. JTAG Connector

Pin/Name	Connected to	Functionality
1-VCC	VCC	-
2-VCC	VCC	-
3-TRST	PIN-20	P1.31/TRST
4-GND	GROUND	-
5-TDI	PIN-60	P1.28/TDI
6-GND	GROUND	-
7-TMS	PIN-52	P1.30/TMS
8-GND	GROUND	-
9-TCK	PIN-56	P1.29/TCK
10-GND	GROUND	-
11-RTCK	PIN-24	P1.26/RTCK
12-GND	GROUND	-
13-TD0	PIN-64	P1.27/TD0
14-GND	GROUND	-
15-RST	PIN-57	RST
16-GND	GROUND	-
19--	NOT CONNECTED	-
20-GND	GROUND	-

3. RS232 Connector 1

Pin/Name	Connected to	Functionality
1-CD	NOT CONNECTED	-
2-RXD	PIN-19	P0.0/TXD0/PWM1
3-TXD	PIN-21	P0.1/RXD0/PWM3/EINT0
4-DTR	SLIDE-SWITCH	-
5-GND	GROUND	-
6-DSR	NOT CONNECTED	-
7-RTS	NOT CONNECTED	-
8-CTS	NOT CONNECTED	-
9-RI	NOT CONNECTED	-

4. RS232 Connector 2

Pin/Name	Connected to	Functionality
1-CD	NOT CONNECTED	-
2-RXD	PIN-33	P0.8/TXD1/PWM4/AD1.1
3-TXD	PIN-34	P0.9/RXD1/PWM6/EINT3
4-DTR	SLIDE-SWITCH	-
5-GND	GROUND	-
6-DSR	NOT CONNECTED	-
7-RTS	NOT CONNECTED	-
8-CTS	NOT CONNECTED	-
9-RI	NOT CONNECTED	-

6.4 Programming

The LPC-2129 can be programmed using Flash utility software provided by Phillips and by H-JTAG software. The program development can be done in the Keil uVision3. For the program development Keil uVision3 must be installed in the computer workstation. Keil platform will compile the files and provide hex files as the output which is to be programmed into the microcontroller. In the present work flash programming is done using Phillips flash utility software. The step by step procedure for the programming is given below.

- Connect Serial cable to COM port of PC
- Connect the other end of Serial cable to Port P2 of ARM7 LPC-2129 board
- Load Phillips flash utility software for programming
- Select COM port number to which the serial cable is connected
- Select the baud-rate as 9600
- Check the check box for DTR and RTS for reset and boot loader selection
- Select the hex file from any of the example code provided by clicking on browse button as shown in the Figure 6-5 below.

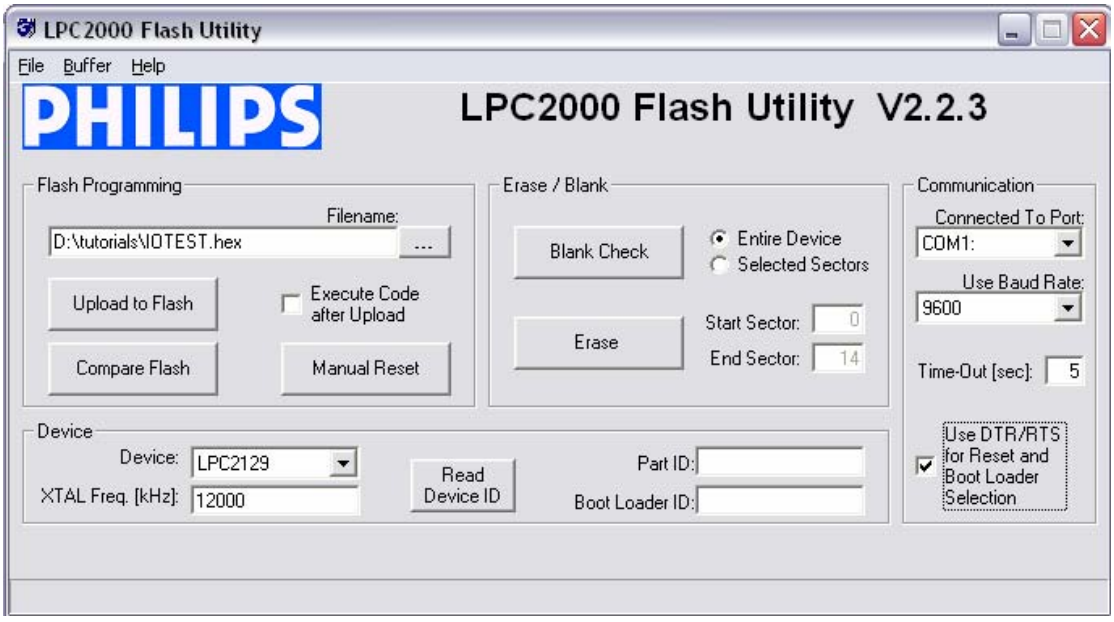


Figure 6-5: Screen shot of the Philips flash utility software.

- After selection click on Read Device ID to detect the target
- After selection click on Upload to Flash and wait till progress bar vanishes. Now the device is programmed and ready for the program run.

Programming Examples

This will clarify the detailed working of the board and the development cycle for the ARM based system design.

1. RS232 Programming

Open the Keil uVision3 from the start menu. The startup screen for the Keil uVision3 is shown in the Figure 6-6.

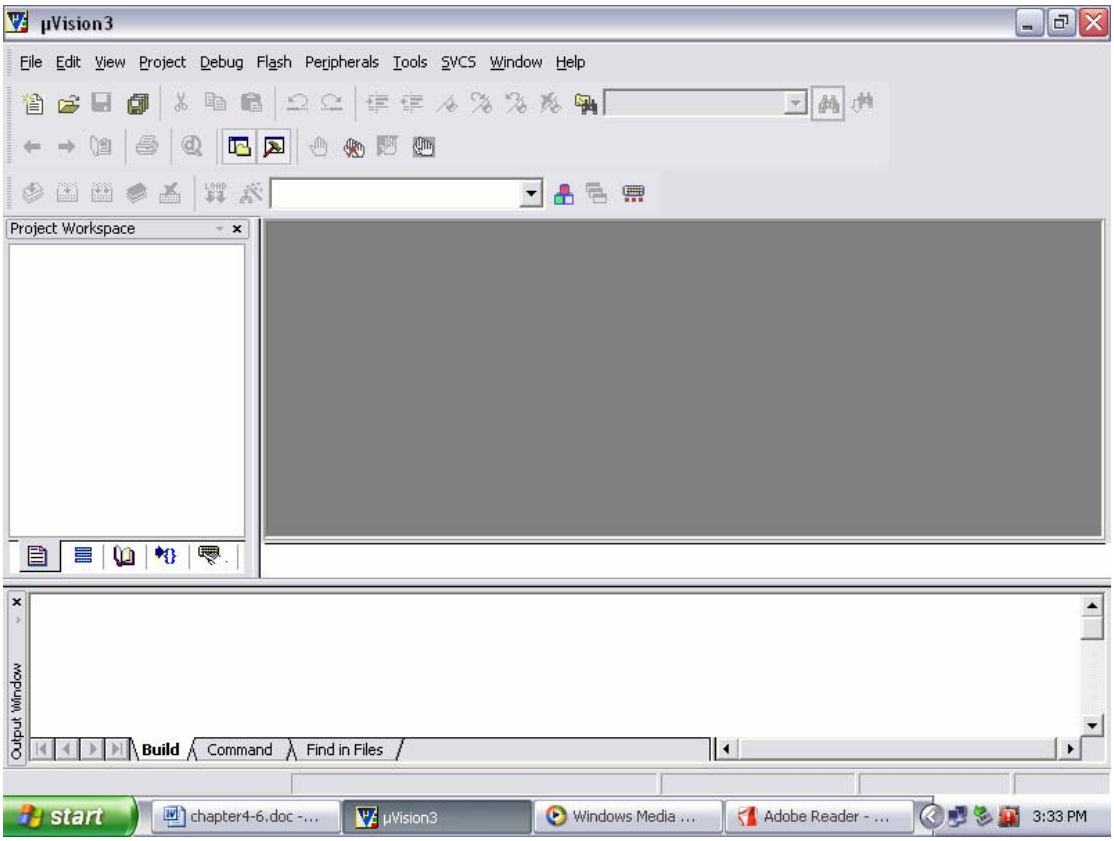


Figure 6-6: Startup screen shot of the Keil uVision3.

Now go to the “Project” menu in the menu bar and click on the “new uVision project” option. This will ask the path where you want to save the project and also to name the project. Give it name ‘serial_test’ and press Save button. This will open a pop-up window to select the device used for the development, in that select the LPC2129 for current case as shown in the Figure 6-7.

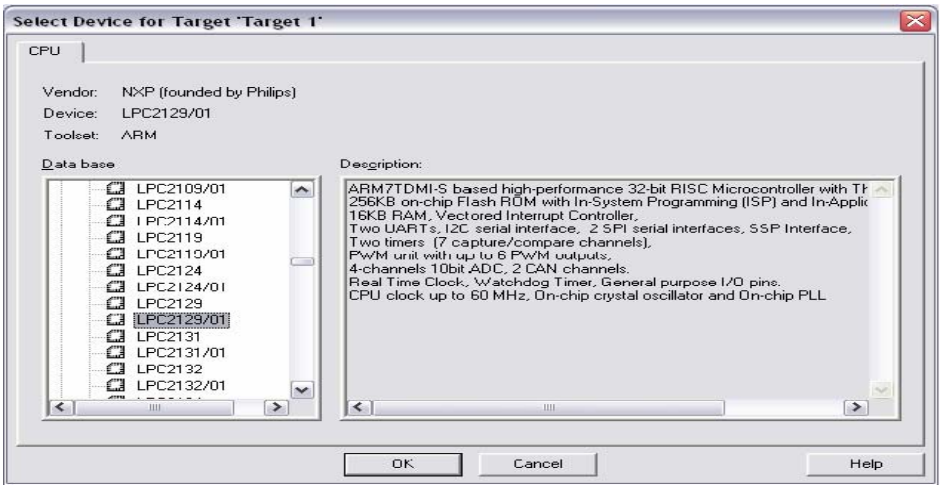


Figure 6-7: Device Select window for the Target.

Press OK after selecting the Target device LPC-2129. This will ask to copy startup codes to the current project, press no in this window. This will open the Keil window as shown in the Figure 6-8.

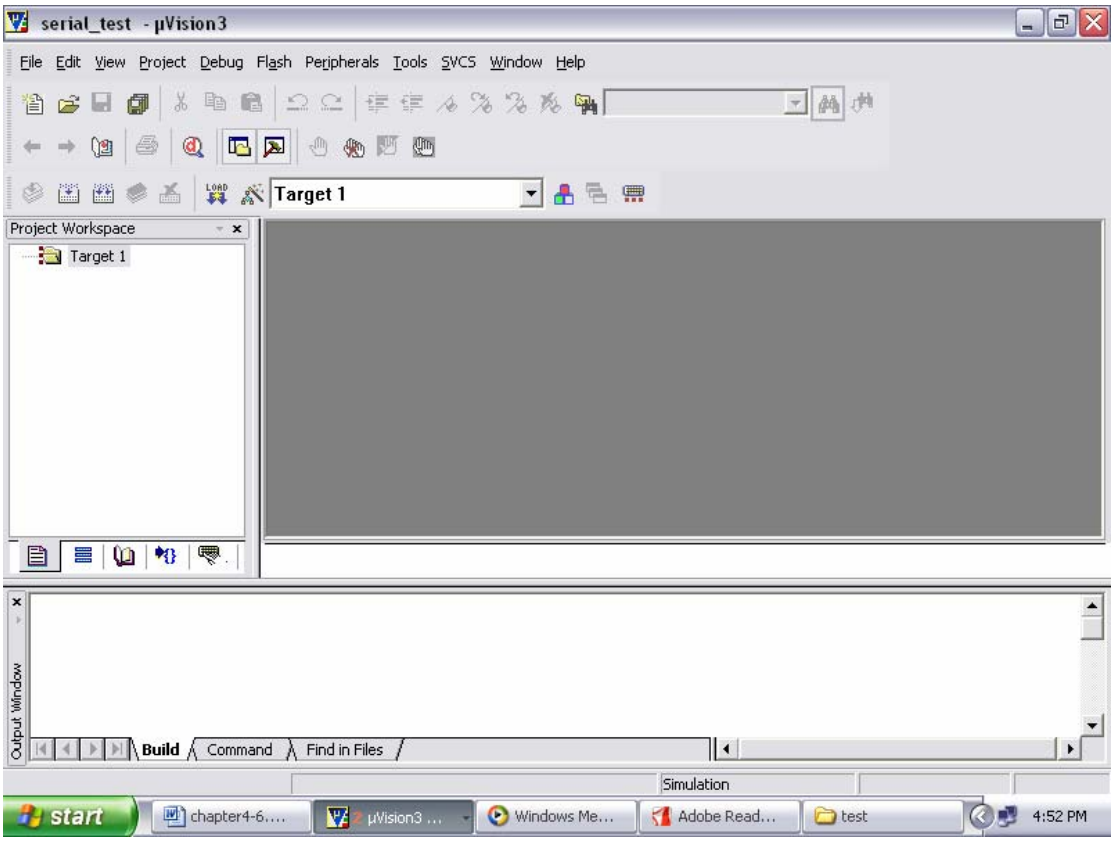


Figure 6-8: Keil uVision3 first workspace screen.

Now enter the program for the RS232 communication between PC and microcontroller using text editor and save it as serial_test.c. Once the program is ready it will be added into the Project Workspace of the Keil uV3 by right clicking the Target 1 shown in the Project Workspace and clicking on the Manage Components as shown in the Figure 6-9 below.

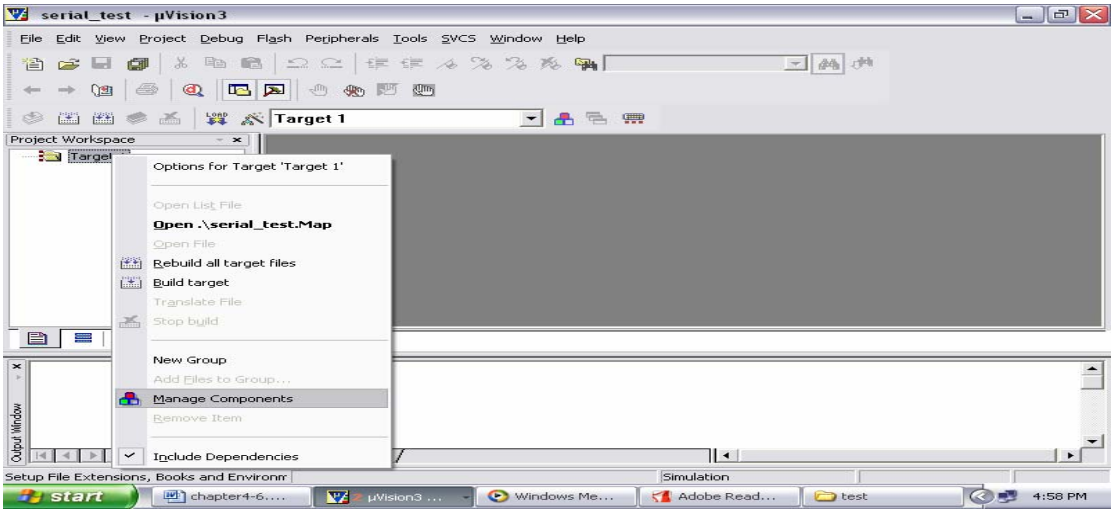


Figure 6-9: Adding files to the project workspace.

This will open a pop-up window as shown in the Figure 6-10.

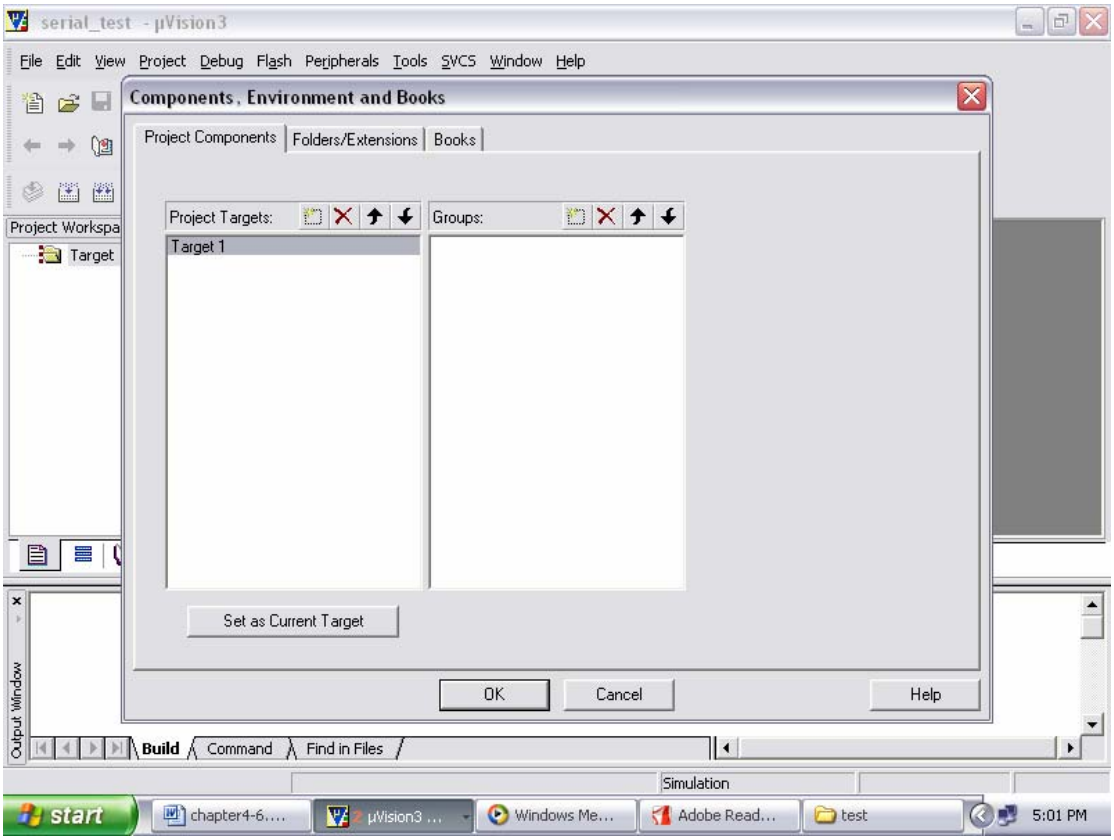


Figure 6-10: Pop window to add groups and files for the project.

In that make two groups Startup Code and Source Code and add files system.s and serial_test.c respectively to the individual group. Press OK this will add the files to the Project work space window as shown in the Figure 6-11.

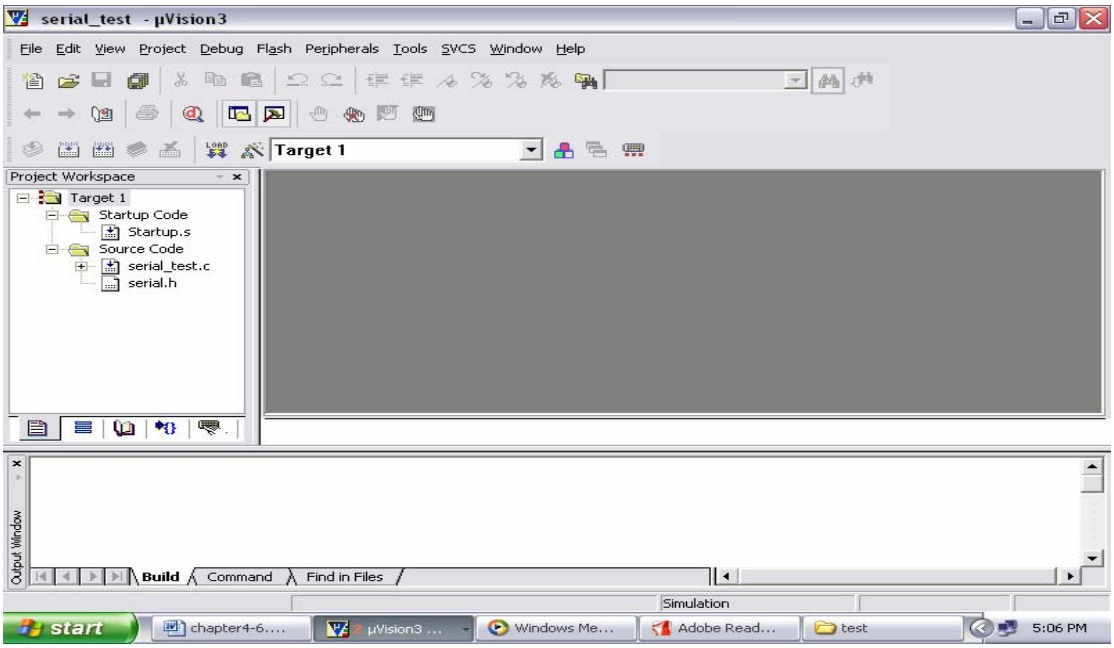


Figure 6-11: Files added to the project work space.

Double clicking the serial_test.c will open the program in the blank space as shown in the Figure 6-12.

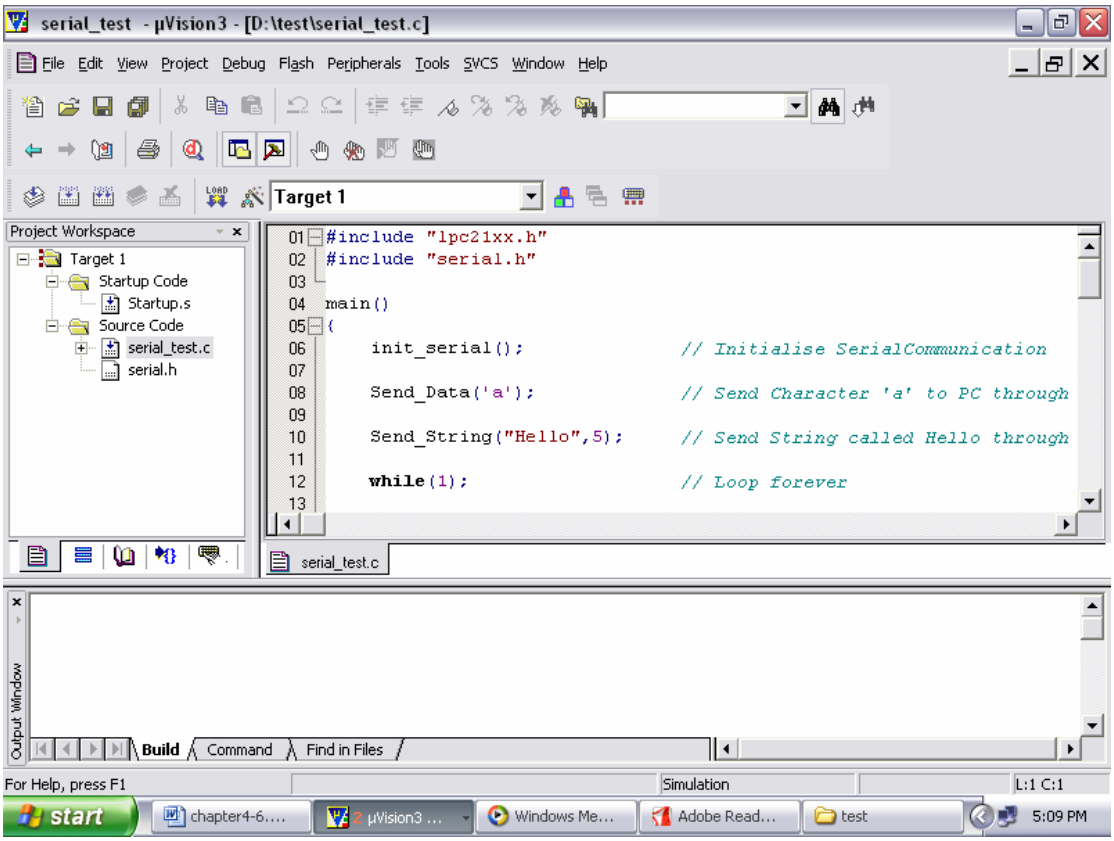


Figure 6-12: Program serial_test.c in the keil editor.

One can do the proper modifications in the program and then build the program. Moreover we are using the Keil compiler to get the hex output file. We need to do some configuration. For this right click on the Target 1

in the project workspace and click on the first option 'Option for the Target'. This will open a popup window as shown in the Figure 6-13.

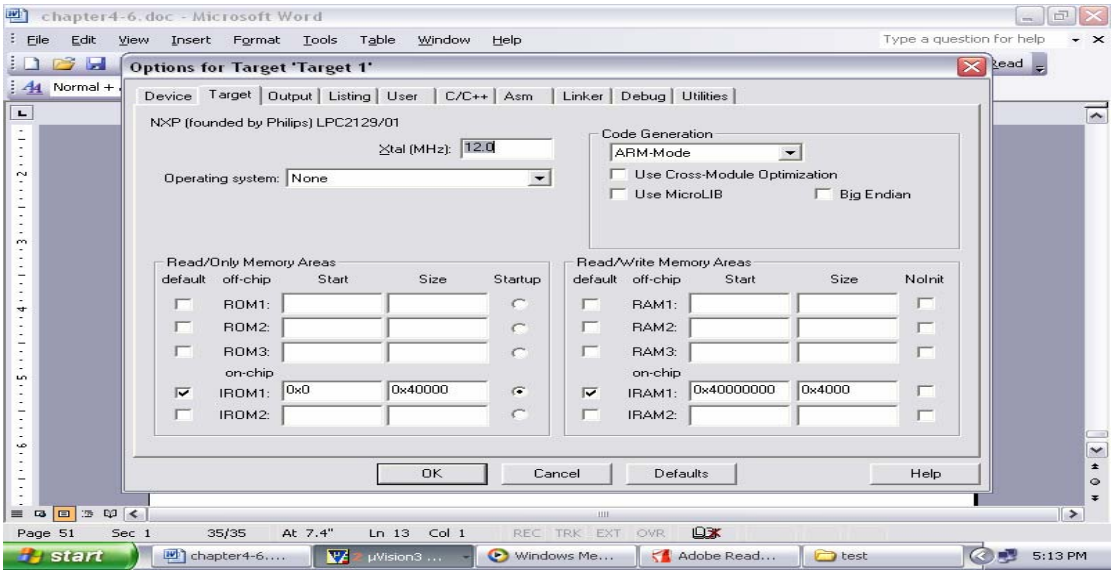


Figure 6-13: Configuration before building the target.

In the output tab click on the Create hex file set box and press OK to come out to the original window. Now go to the project menu and build the target. This will create hex file at the same path where the serial_test.c file is saved. The final screen of the Keil IDE shows the message in the bottom for the successful compilation and creation of the hex file as shown in the Figure 6-14.

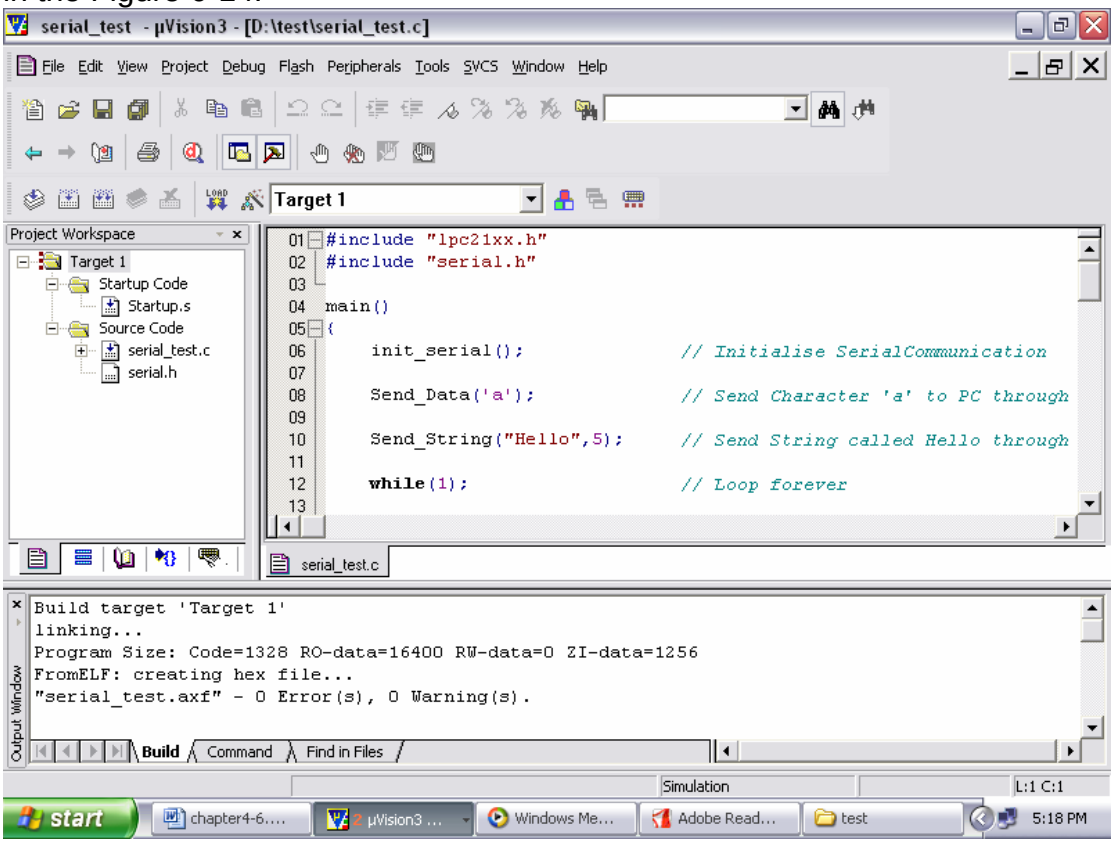


Figure 6-14: Final Screen shot of the Keil IDE.

The hex file is now ready to be downloaded in to the flash of the LPC2129 microcontroller using Phillips flash utility software. One can follow the same procedure for any program development and synthesis.

The serial_test.c program is to send “Hello” string to the PC serial port and is listed below. It can be checked by the windows HyperTerminal for its proper working after downloading the serial_test.hex file into the microcontroller memory.

```
#include "lpc21xx.h"
#include "serial.h"

void main()
{
    init_serial();                  // Initialise SerialCommunication

    Send_String("Hello",5);    // Send String called Hello through Serial Port

    while(1);                       // Loop forever
}
void init_serial (void)  {
  PINSEL0 = 0x00000005;              /* Enable RxD1 and TxD1 */
  U0LCR = 0x83;                    /* 8 bits, no Parity, 1 Stop bit */
  U0DLL = 97;                     /* 9600 Baud Rate @ 15MHz VPB Clock */
  U0LCR = 0x03;                    /* DLAB = 0 */
}
int Send_Data (int data)
{

  while (!(U0LSR & 0x20));
  U0THR = data;
  return (U0THR);

}

int Receive_Data (void)
{

  while (!(U0LSR & 0x01));

  return (U0RBR);

}

void Send_String(char* string, unsigned char no_of_bytes)
{

    int i, stringlength = no_of_bytes;

    for(i=1; i<=stringlength; i++)
```

```
        {
                Send_Data(*string++);
        }

    }
```

The header file serial.h is also listed below.

```
#ifndef __REG51_H__
#define __REG51_H__

void init_serial (void);
int Send_Data (int ch);
int Receive_Data (void);
void Send_String(char* string, unsigned char no_of_bytes);

#endif
```

This program prints the “Hello” string on the PC HT screen when the LCP2129 board is running in the user mode and its port1 is connected to the PCs RS232 DB9 port.

2. GPIO Test

This program will check for the proper working of the boards GPIO P1.16 to P1.23 as the eight LEDs which are provided on-board are connected with them. The program written in C language is listed below. One will follow the same steps as discussed previously for the hex file creation and downloading it into microcontroller memory.

```
#include <LPC21xx.H>                    /* LPC21xx definitions  */

void ledonoff(void);
void delay(void);

int main()
{
        IODIR1 = 0x00ff0000;          // Make port bits P1.16.....P1.23 as output
        ledonoff();                    // Initialize LED
   return 0;

}

void ledonoff(void)
{
        while(1)
        {
                IOSET1 = 0x00ff0000;       //Make port bits high
                delay();
```

```
                IOCLR1 = 0x00ff0000;       //Make port bits low
                delay();
        }

}

void delay(void)
{
        int i;
        char k;
        for(k=0;k<=8;k++)
        {
                for(i=0;i<=100000;i++) ;
        }
}
```

This will blink eight LEDs of the LPC-2129 board at a delay set in the program above.

Conclusion

A prototype board for the basic 32-bit ARM7 microcontroller is developed and experimented in the research laboratory. This will definitely provide a perfect base for the better embedded system design and applications. For future developments, I would like to provide the CAN interface connectors to the board and might switch to advanced ARM architectures after having hands on to the basic ARM7 architecture.

Section - III: Design of the microcontroller based interfacing modules and Application.

Section 3: Design of the microcontroller based interfacing modules and application.

Chapter 7 LCD Interface: An Electronic Name Plate

7.1 Introduction

Sometimes it becomes necessary to display complex message which can not be tackled by simple LED's or 7-segment displays. Such display messages could be made up of numbers, characters of the alphabet, and other symbols. This type of display messages can be handled by different types of LCD modules. In this chapter LCD interface with the microcontroller and a simple application an electronic name plate is presented.

Figure 7-1: Electronic Name-Plate cum Show Piece.

An LCD module contains one or more rows of character positions. Each character position consists of a matrix that is typically five segments, or dots wide and eight segments tall. The module forms characters by turning on the appropriate segments in a character position. LCD's are available in several sizes: 1x16(1 line of 16 characters), 2x16, and 2x20 are some popular sizes used in products.

A photograph of electronic name plate cum show piece is shown in Figure 7-1 along with the circuit board. Table 7-1 summarizes the signals in the 16-line interface in 1x16 LCD modules.

PIN	SYMBOL	INPUT/OUTPUT	FUNCTION
1	GND	INPUT	SIGNAL GROUND
2	VDD	INPUT	SUPPLY VOLTAGE (+5V)
3	V0	INPUT	CONTRAST ADJUST
4	RS	INPUT	REGISTER SELECT (1=DATA; 0=INSTRUCTION REGISTER, BUSY FLAG/ADDRS COUNTER)
5	R/W	INPUT	READ (1)/WRITE(0) SELECT
6	E	INPUT	ENABLE
7	DB0	INPUT/OUTPUT	DATA BIT 0
8	DB1	INPUT/OUTPUT	DATA BIT 1
9	DB2	INPUT/OUTPUT	DATA BIT 2
10	DB3	INPUT/OUTPUT	DATA BIT 3
11	DB4	INPUT/OUTPUT	DATA BIT 4
12	DB5	INPUT/OUTPUT	DATA BIT 5
13	DB6	INPUT/OUTPUT	DATA BIT 6
14	DB7	INPUT/OUTPUT	DATA BIT 7
15	BL1	INPUT	BACKLIGHT (ANODE)
16	BL2	INPUT	BACKLIGHT (CATHODE)

Table 7-1: Signals of 1x16 LCD module

LCD modules use backlighting to allow viewing in dim light. A module may be reflective (which does not use backlight), Transmissive (which must use backlight), or transflective (which may use backlight or not).

7.2 About LCD Modules

LCD module is a specialized microcontroller in itself. It contains its own RAM and ROM, and executes the instructions shown below in Table 7-2.

NO.	INSTRUC-TION	RS	R/W	D7	D6	D5	D4	D3	D2	D1	D0	FUNCTION
1	DISPLAY CLEAR	0	0	0	0	0	0	0	0	0	1	CLEAR DISPLAY, RESET DISPLAY FROM SHIFT, SET DDRAM=0
2	CURSOR HOME	0	0	0	0	0	0	0	0	1	X	SHIFT=0, DDRAM=0
3	ENTRY MODE SET	0	0	0	0	0	0	0	1	I/D	S	I/D: INCREMENT(1), DECREMENT(0) CURSOR OR DISPLAY SHIFT AFTER DATA TRANSFER
4	DISPLAY ON/OFF	0	0	0	0	0	0	1	D	C	B	D: DISPLAY ON (1), OFF (0) C: CURSOR ON (1), OFF (0) B: CURSOR BLINK ON (1), OFF (0)

5	DISPLAY/ CURSOR SHIFT	0	0	0	0	0	1	S/C	R/L	X	X	S/C: SHIFT DISPLAY (1), CURSOR (0) R/L: SHIFT RIGHT (1), LEFT (0)
6	FUNCTION SET	0	0	0	0	1	DL	N	0	X	X	DL: 8-BIT (1), 4-BIT (0) INTERFACE N: DUAL (1), SINGLE (0) LINE DISPLAY
7	CG RAM ADDRS SET	0	0	0	1	CG 5	CG 4	CG 3	CG 2	CG 1	CG 0	LOAD ADDRS COUNTER WITH CG0-CG5 SUBSEQUENT DATA GOES TO CGRAM
8	DD RAM ADDRS SET	0	0	1	DD 6	DD 5	DD 4	DD 3	DD 2	DD 1	DD 0	LOAD ADDRS COUNTER WITH DD0 TO DD6 SUBSEQUENT DATA GOES TO DDRAM
9	BUSY FLAG/AD--DRS COUNTE--R READ	0	1	BF	AC 6	AC 5	AC 4	AC 3	AC 2	AC 1	AC 0	READ BUSY FLAG (BF) AND 0 ADDRS COUNTER (AC0-AC6)
10	CG/DD RAM DATA WRITE	1	0	D7	D6	D5	D4	D3	D2	D1	D0	WRITE DATA (D0-D7) TO CGRAM OR DDRAM
11	CG/DD RAM DATA READ	1	1	D7	D6	D5	D4	D3	D2	D1	D0	PLACE DATA FROM CGRAM OR DDRAM ON D0-D7

Table 7-2: Instruction table.

7.3 Memory Areas Inside LCD

The modules on chip memory include a CG (character generator) ROM, CGRAM, DD (display data) RAM, an instruction register, and a data register. The CGROM stores patterns for generating 192 different characters. These are fixed in ROM and can not be altered. The CGRAM stores segment patterns for up to 16 user designed characters such as logos, special symbols, or other simple graphics characters that we design on the 5x8 matrix. To create a custom character, we write a series of 5-bit words to the CGRAM. Each word represents the segment pattern for one row in the desired character. The patterns stored in CGRAM disappear on powering down, so we must reload them on each time we power up. Each character in the CGROM and CGRAM has an 8-bit address, or character code. Conveniently, the codes for the upper and lower case Roman alphabet and common punctuation are same as the ASCII codes for those characters (21h

through 7dh). For example, the pattern for A is stored at address 41h, B is stored at 42h, and so on. An 8-bit instruction register (IR) stores instruction code and addresses, and an 8-bit data register (DR) stores character codes. When we read or write to the chip, we must select the appropriate register. The DDRAM stores up to eighty 8-bit character codes each character position on the display corresponds to an address in the DDRAM, and the character codes stored in the DDRAM determine what is displayed at each position.

Display position DDRAM address is shown below in Table 7-3 for the LCD module.

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
00h	01h	02h	03h	04h	05h	06h	07h	40h	41h	42h	43h	44h	45h	46h	47h

Table 7-3: Addresses for display position.

On power on the LCD DDRAM is set to 00h, so we can directly write data to first 8-bits from left side as shown in Table 7-3. However, the 9th character as shown in memory map is at address 40h. This means that if we write a character after 8th character directly it will not appear on 9th character position. That is because the 9th character will effectively be written to address 09h, but the 9th character position is at address 40h.
Thus we need to send command to the LCD that tells it to position the cursor on the 9th character position. The “set cursor position” instruction is 80h. To do this we must add the address of the location where we wish to position the cursor. So to display character at 9th position we must add 80h + 09h = 0C0h. Thus sending 0C0h to the LCD will position the cursor on 9th character position of the LCD and write the user defined character.

7.4 Interfacing LCD Module with microcontroller

Figure 7-2, shows the schematic diagram of the interface module.

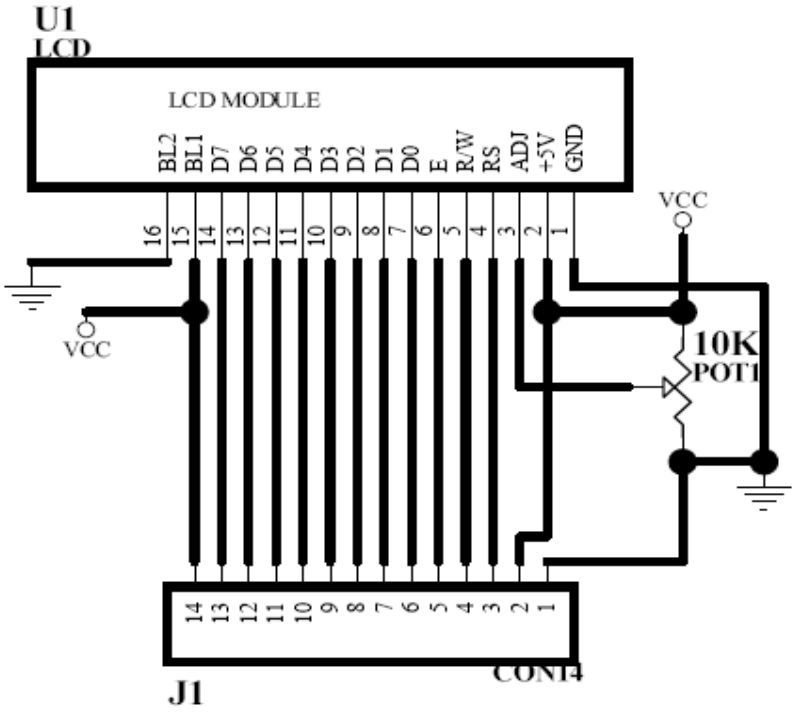


Figure 7-2: Schematic diagram of interface module.

All required signals are taken out to connector J1. Pin configuration of J1 is as follows:

PIN NO.	SIGNAL
1	GND
2	VCC
3	RS
4	R/W
5	E
6	D0
7	D1
8	D2
9	D3
10	D4
11	D5
12	D6
13	D7
14	VCC

Note that BL1 (anode) connected to VCC and BL2 (cathode) connected to GND and 10k pot is connected to set the contrast of the LCD.

Connection Details of LCD Module with AT89c51 Microcontroller Ports:

Signal from J1	AT89c51 Port Pins
GND	GND
VCC	VCC
RS	P3.2
R/W	P3.3
E	P3.4
D0	P1.0
D1	P1.1
D2	P1.2
D3	P1.3
D4	P1.4
D5	P1.5
D6	P1.6
D7	P1.7

7.5 Programming the Microcontroller

Any flash microcontroller can be used for this application. As the programmer for AT89c51 is available now it is good to program it using any serial or parallel programmer. Using assembler (ASM51) one can easily change *.asm extension file to *.hex (machine codes) which can be burnt in to the microcontroller.

Program given here will Display “Dr.K.P.JOSHIPURA” first and then it is cleared to display “VICE-CHANCELLOR”, and keeps on displaying this two words one by one continuously.

7.6 Software Flowchart and Description

Flowchart for the logical understanding of microcontroller LCD module interface is given in Figure 7-3.

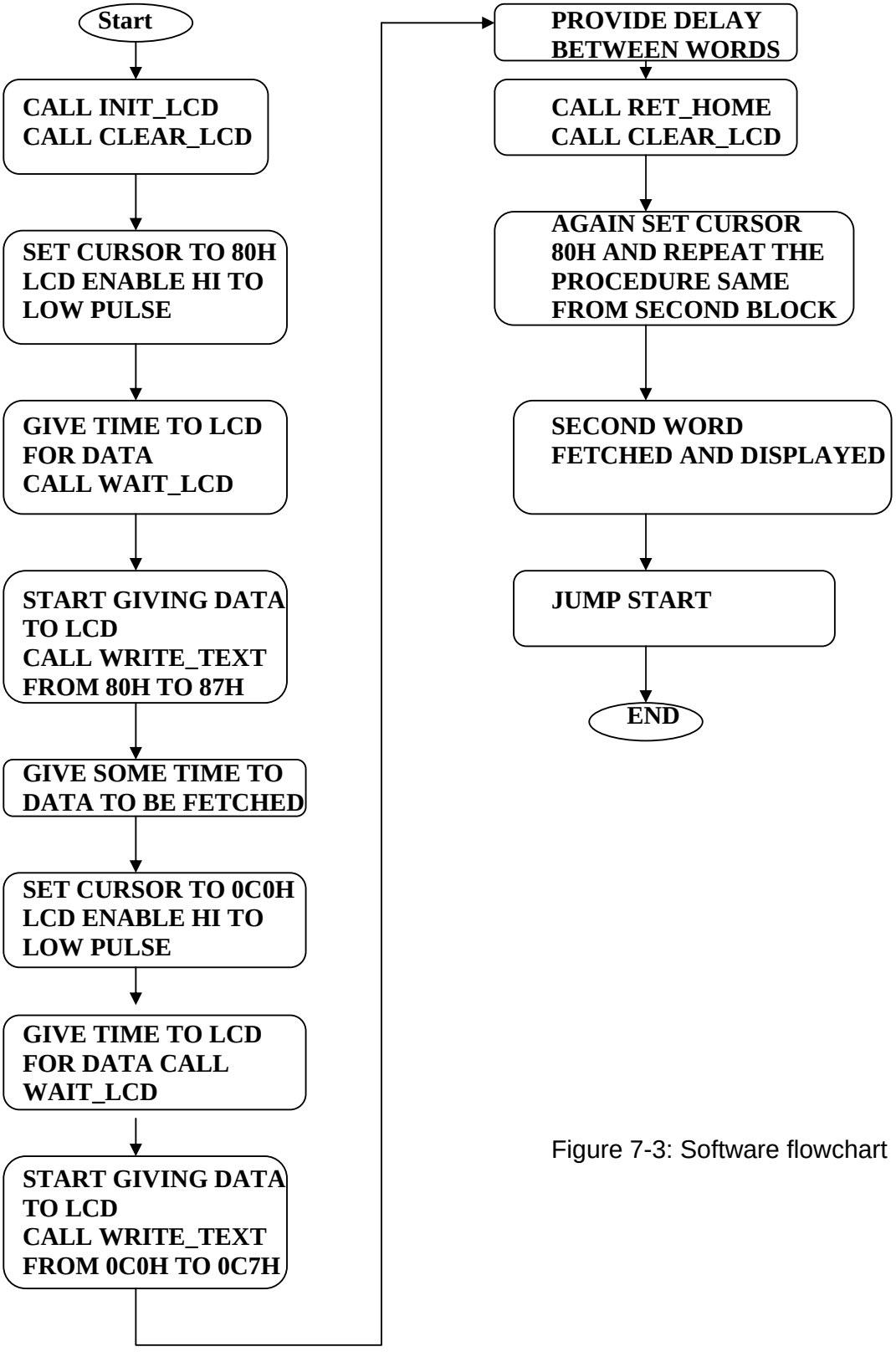


Figure 7-3: Software flowchart

Software for LCD initialization and data write is self explanatory and is given as below:

LCD Module Software

```
; Name plate using LCD
; LCD initialization and DATA1 write program
; Main program
$MOD51
        DB0     EQU     P1.0        ; Equate P1, P1.0 to P1.7
        DB1     EQU     P1.1        ; to DB0-DB7
        DB2     EQU     P1.2
        DB3     EQU     P1.3
        DB4     EQU     P1.4
        DB5     EQU     P1.5
        DB6     EQU     P1.6
        DB7     EQU     P1.7
        EN      EQU     P3.4        ; Equate P3, P3.2 to P3.4
        RW      EQU     P3.3        ; to EN, RW, RS respectively
        RS      EQU     P3.2
        DATA1   EQU     P1          ; Equate P1 to DATA1
                AJMP    MAIN
                ORG     130H        ; Start program from 130h
MAIN:
        MOV     SP, #50H            ; Move stack pointer to 50h
        LCALL   INIT_LCD            ; Call initialization codes
        LCALL   CLEAR_LCD           ; Call clear LCD codes
        CLR     RS                  ; Set cursor to 00h of LCD
        MOV     DATA1, #80H         ; Means 80h+00h=80h
        SETB    EN                  ; HI to LO pulse
        NOP
        NOP
        NOP
        NOP
        NOP
        NOP
        NOP
        CLR     EN
        LCALL   WAIT_LCD            ; Give time to LCD for data
        MOV     A, #'D'             ; Start giving data to LCD
        LCALL   WRITE_TEXT          ; to be displayed from 80h
        MOV     A, #'r'
        LCALL   WRITE_TEXT          ; Writes data to LCD
        MOV     A, #'.'
        LCALL   WRITE_TEXT
        MOV     A, #'K'
        LCALL   WRITE_TEXT
        MOV     A,  #'.'
        LCALL   WRITE_TEXT
```

```
MOV     A, #'P'
LCALL   WRITE_TEXT
MOV     A, #'.'
LCALL   WRITE_TEXT
MOV     A, #'J'                 ; Cursor position at 87h
LCALL   WRITE_TEXT
NOP
NOP
NOP
NOP
NOP
NOP
NOP
CLR     RS
MOV     DATA1, #0C0H            ; move cursor to 0C0h
SETB    EN                      ; i.e. 80h+40h=0C0h
NOP
NOP
NOP
NOP
NOP
NOP
NOP
CLR     EN
LCALL   WAIT_LCD
MOV     A, #'O'                 ; Start giving data to Display
LCALL   WRITE_TEXT
MOV     A, #'S'
LCALL   WRITE_TEXT
MOV     A, #'H'
LCALL   WRITE_TEXT
MOV     A, #'I'
LCALL   WRITE_TEXT
MOV     A, #'P'
LCALL   WRITE_TEXT
MOV     A, #'U'
LCALL   WRITE_TEXT
MOV     A, #'R'
LCALL   WRITE_TEXT
MOV     A, #'A'
LCALL   WRITE_TEXT
LCALL   DEL1                    ; Stay here for a while
LCALL   DEL1
LCALL   DEL1
LCALL   RET_HOME                ; Again back to 80h
LCALL   CLEAR_LCD               ; Clear LCD cursor on Right
CLR     RS
MOV     DATA1, #80H             ; Set cursor to 80h
SETB    EN
NOP
```

```
NOP
NOP
NOP
NOP
NOP
NOP
CLR     EN
LCALL   WAIT_LCD
MOV     A, #'V'                 ;Display data
LCALL   WRITE_TEXT
MOV     A, #'I'
LCALL   WRITE_TEXT
MOV     A, #'C'
LCALL   WRITE_TEXT
MOV     A, #'E'
LCALL   WRITE_TEXT
MOV     A, #'-'
LCALL   WRITE_TEXT
MOV     A, #'C'
LCALL   WRITE_TEXT
MOV     A, #'H'
LCALL   WRITE_TEXT
MOV     A, #'A'
LCALL   WRITE_TEXT
NOP
NOP
NOP
NOP
NOP
NOP
NOP
CLR     RS
MOV     DATA1, #0C0H            ; Set cursor to 0C0h
SETB    EN
NOP
NOP
NOP
NOP
NOP
NOP
NOP
CLR     EN
LCALL   WAIT_LCD
MOV     A, #'N'                 ; Display DATA
LCALL   WRITE_TEXT
MOV     A, #'C'
LCALL   WRITE_TEXT
MOV     A, #'E'
LCALL   WRITE_TEXT
MOV     A, #'L'
```

```
        LCALL   WRITE_TEXT
        MOV     A, #'L'
        LCALL   WRITE_TEXT
        MOV     A, #'O'
        LCALL   WRITE_TEXT
        MOV     A, #'R'
        LCALL   WRITE_TEXT
        MOV     A, #' '
        LCALL   WRITE_TEXT
        LCALL   DEL1
        LCALL   DEL1
INIT_LCD:                       ; LCD ready subroutine
        CLR     RS              ; RS=0
        MOV     DATA1, #38H     ; Function set for 1line, 5x8
        SETB    EN              ; HI to LO pulse
        NOP
        NOP
        NOP
        NOP
        NOP
        NOP
        NOP
        CLR     EN
        LCALL   WAIT_LCD        ; Check for data fetch
        CLR     RS              ; RS=0
        MOV     DATA1, #0EH     ; Display on, cursor on cmd
        SETB    EN              ; HI to LO pulse
        NOP
        NOP
        NOP
        NOP
        NOP
        NOP
        NOP
        CLR     EN
        LCALL   WAIT_LCD        ; Check for data fetch
        CLR     RS              ; RS=0
        MOV     DATA1, #06H     ; Entry mode set for LCD
        SETB    EN              ; HI to LO pulse
        NOP
        NOP
        NOP
        NOP
        NOP
        NOP
        NOP
        CLR     EN
        LCALL   WAIT_LCD        ; Check for data fetch
        RET                     ; Return to main program
CLEAR_LCD:
```

```
        CLR     RS                  ; RS=0
        MOV     DATA1, #01H         ; Clear LCD cmd
        SETB    EN                  ; HI to LO pulse
        NOP
        NOP
        NOP
        NOP
        NOP
        NOP
        NOP
        CLR     EN
        LCALL   WAIT_LCD            ; Check for data fetch
        RET                         ; Return to main program
WRITE_TEXT:                         ; Write DATA to LCD
        SETB    RS                  ; RS=1
        MOV     DATA1 ,A            ; Move desired char to P1
        SETB    EN                  ; HI to LO pulse
        NOP
        NOP
        NOP
        NOP
        NOP
        NOP
        NOP
        CLR     EN
        LCALL   WAIT_LCD            ; Check for data fetch
        RET                         ; Return to main program
WAIT_LCD:                           ; Check for data fetch
        MOV     R2, #03H            ; Count in R2
        MOV     R3, #0FFH           ; Count in R3
        CLR     EN                  ; EN=0, RS=0, RW=1
        CLR     RS
        SETB    RW
        MOV     DATA1, #0FFH        ; P1 as input port
        SETB    EN                  ; EN=1
        MOV     A, DATA1            ; Get data from P1
LOOP:
        JB      ACC.7, HERE2        ; Check for DB7 Bit
HERE2:
        NOP                         ; High=check more
        NOP
        DJNZ    R3, HERE2
        DJNZ    R2, LOOP
        CLR     EN                  ; Otherwise EN=0,
        CLR     RW                  ; RW=0
        LCALL   DELAY               ; Provide delay
        RET                         ; Return to main program
RET_HOME:
        CLR     RS                  ; RS=0
        MOV     DATA1 ,#20H         ; Return home cmd
```

```
        SETB    EN                  ; HI to LO pulse
        NOP
        NOP
        NOP
        NOP
        NOP
        NOP
        NOP
        CLR     EN
        LCALL   WAIT_LCD            ; Check for data fetch
        RET                         ; Return to main program
DELAY:
        MOV     R2, #03             ; More than 2ms delay
AGAIN:
        MOV     R3, #250
HERE1:
        NOP
        NOP
        DJNZ    R3, HERE1
        DJNZ    R2, AGAIN
        RET                         ; Return to main program
DEL1:
        MOV     89H, #10H           ; More than 2s delay
        MOV     R5, #25
AGAIN1:
        MOV     8BH, #00H
        MOV     8DH, #00H
        MOV     88H, #60H
PON:    JNB     8FH, PON
        MOV     88H, #00H
        DJNZ    R5, AGAIN1
        RET                         ; Return to main program
        END                         ; End of Program
```

Component layout diagram

Figure 7-4 shows the top-side component layout diagram for the LCD microcontroller interface.

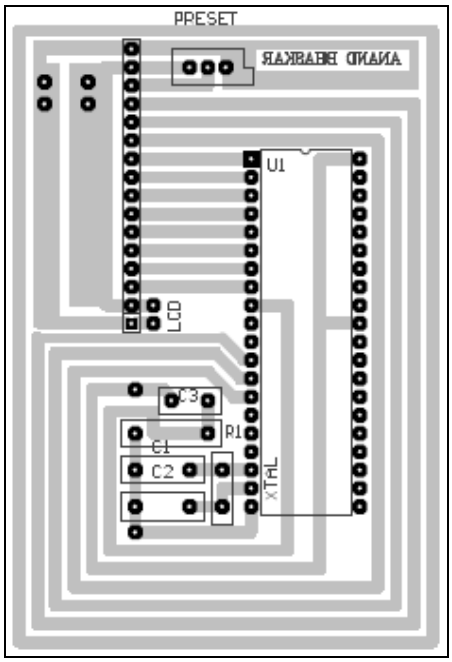


Figure 7-4: Component layout diagram.

Solder Side PCB for LCD Microcontroller Interface

The solder side PCB for the electronic name plate is shown in the Figure 7-5.

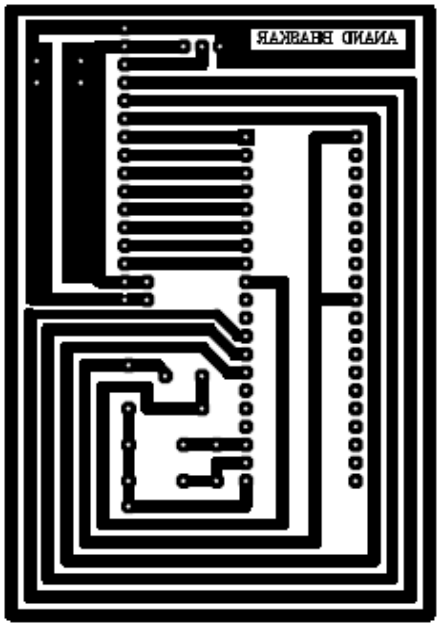


Figure 7-5: Solder side (bottom) PCB.

7.7 Future Developments

LCD interface with microcontroller definitely provides one of the good perspectives for better output device which gives user friendly environment to the microcontroller based embedded systems. Many dedicated applications can employ an LCD interface. Here only one of the LCD modules is used for application development but many power efficient and less wire LCD module interfaces can be worked out as a part of future developments for efficient embedded system design using microcontroller.

Chapter 8 LCD-Keyboard Interface

A Keyboard Interface with microcontroller system is discussed in this chapter. The system includes AT89C51 Microcontroller, 40-key Matrix Keyboard, and 1-Line 16-Character LCD Module. The designing, circuit explanation and software development is presented.

8.1 Introduction

Keyboards offer more options than individual switches or pushbuttons, at lower cost and compact size. It is the basic input device for any system by which human can easily communicate with machine. There are many applications in which keyboards are used, like Electronic Locks, EPROM Programmers, and many Test Instruments.

In present work, an 8x5 matrix is used as a 4x10 matrix keyboard and interfaced with AT89C51 microcontroller. Here we are focusing more on keyboard interface rather than LCD as it is already done in the previous chapter.

8.2 Basic Understanding of the 40-key Matrix Keyboard

The present work is centered on the interfacing of microcontroller AT89C51 and Keyboard. We have considered here the 8x5 matrix keyboard and arranged the 8x5 matrix into 4x10 matrixes. This arrangement will provide a compact 40-key 4x10 matrix keyboard which can be easily implemented to any application.

Figure 8-1 shows the 8x5 matrix arrangement. Figure 8-2 shows the 4x10 matrix arrangement for the 40-key keyboard.

Thirteen control lines are shown in Figure-2, which are interfaced with the microcontroller. Each key is connected to its corresponding two control lines, one row side and one column side. These thirteen control lines are divided into row and column signals. One to Five control lines are controlling the columns and Six to Thirteen are for the rows. Compare this with Figure-1. See the numbers like 1-13, 1-12, 2-13 …. , 5-8 shown in Figure-2. These numbers show that when the user press the key numbered 1-13 for example, the control lines 1 & 13 are shorted and show continuity. One can check the working of keyboard by pushing each key and measuring the continuity between numbered control lines.

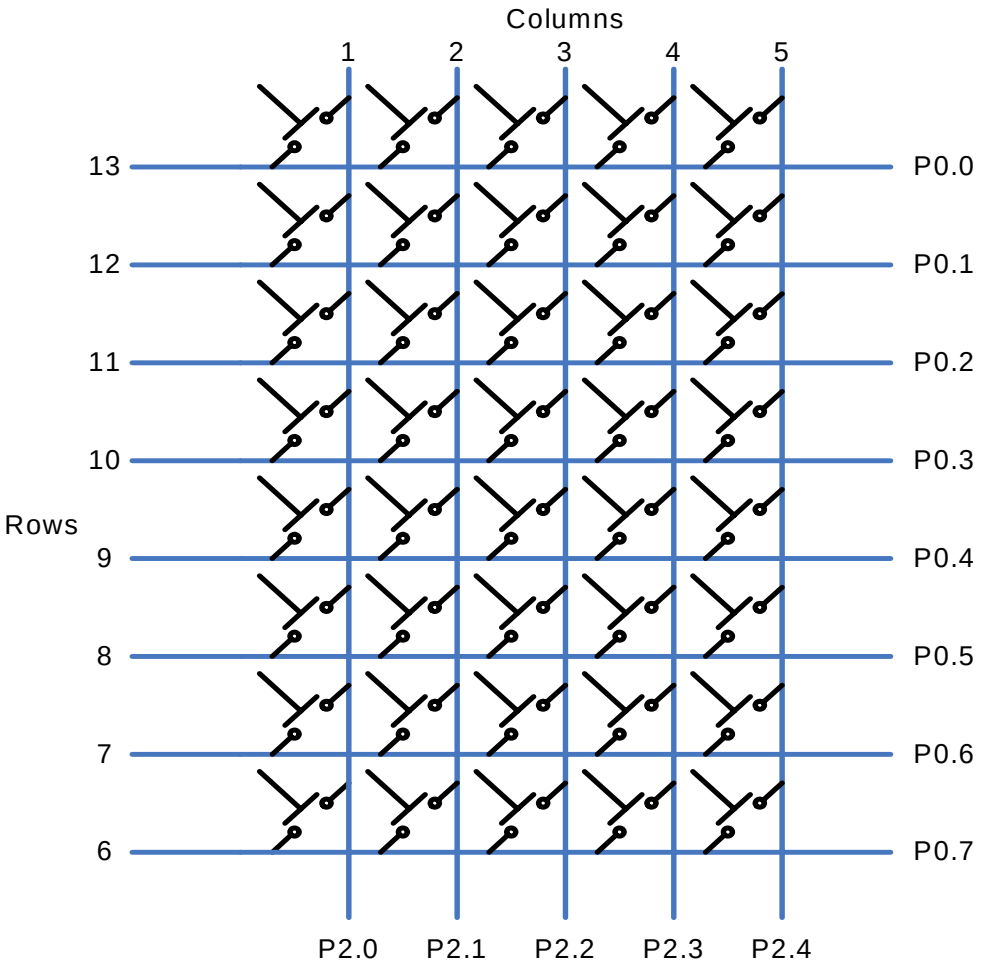
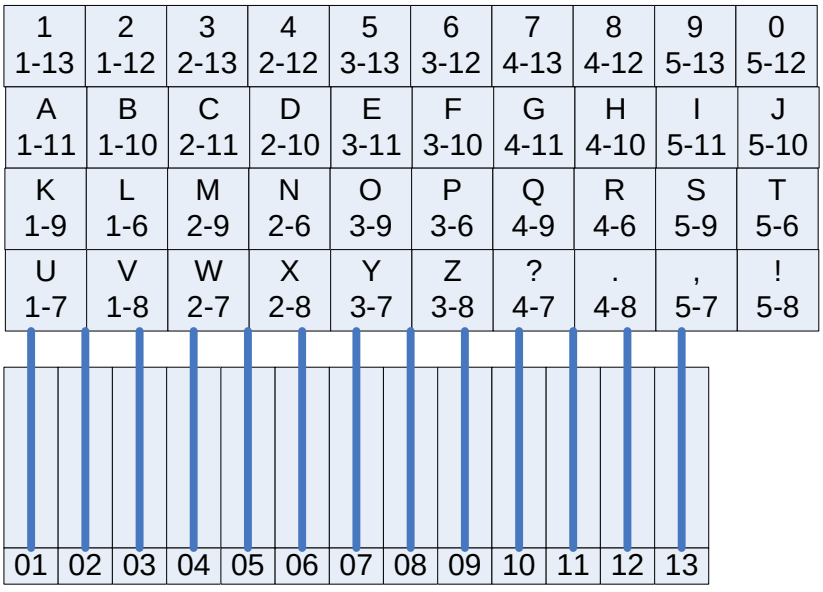


Figure-1: The 8x5 matrix arrangement of the Keyboard.



Control Lines to Microcontroller

Figure-2: 4x10-Matrix for 40-key Keyboard.

8.3 Detailed Circuit Diagram and Explanation

To see the result of key pressed we have used the 1-line 16-character LCD module. The details of the circuit diagram are shown in Figure 8-3.

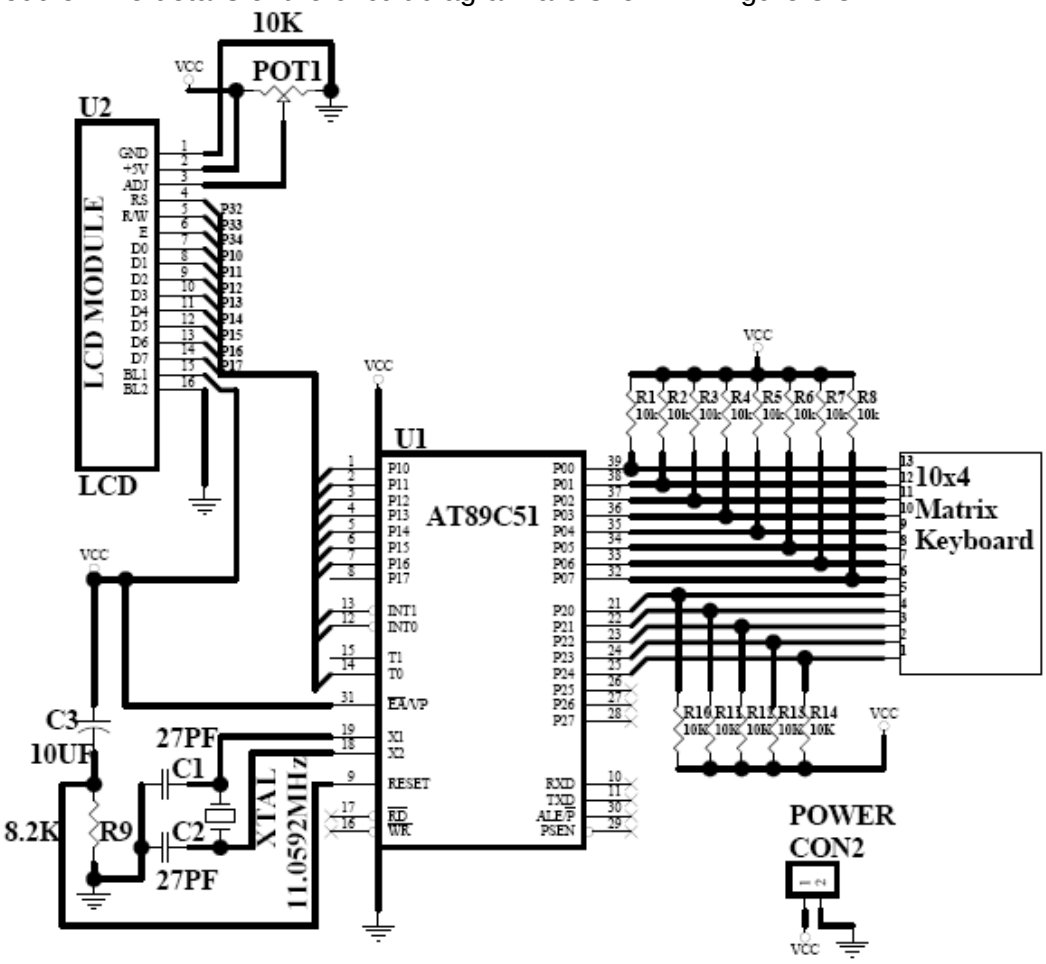


Figure 8-3: Circuit diagram of keyboard interface with AT89C51 microcontroller.

We have connected each control line of the keyboard to VCC via 10K resistor. Figure 8-4 shows the logic for single key of the keyboard. Initially the rows and columns are getting +5v.

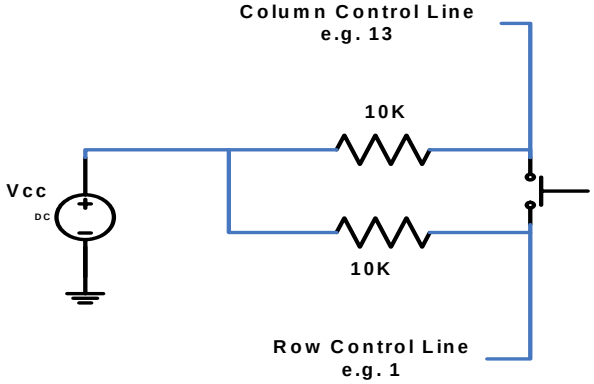


Figure 8-4: The logic for single key of the keyboard.

The columns control lines (1-5) are connected to Port 2 (P2.0-P2.4) and the row control lines (6-13) are connected to Port 0 (P0.0 to P0.7). On the other side of AT89C51 microcontroller LCD module is interfaced with Port 1 (P1.0-P1.7) and Port 3 (P3.2, P3.3, & P3.4). The connection details of the Keyboard & LCD module are listed in Table 8-1(a) and Table 8-1(b).

Signal from LCD module	AT89c51 Port Pins
GND	GND
VCC	VCC
RS	P3.2
R/W	P3.3
E	P3.4
D0	P1.0
D1	P1.1
D2	P1.2
D3	P1.3
D4	P1.4
D5	P1.5
D6	P1.6
D7	P1.7

Table 8-1(a): Connection details of LCD and AT89C51 microcontroller.

Control Lines		AT89c51 Port Pins
ROWS	13	P0.0
	12	P0.1
	11	P0.2
	10	P0.3
	09	P0.4
	08	P0.5
	07	P0.6
	06	P0.7
COLUMNS	05	P2.4
	04	P2.3
	03	P2.2
	02	P2.1
	01	P2.0

Table-1(b): Connection details of Keyboard and AT89C51 microcontroller.

Circuit Function

The 10x4 matrix keyboard shown in Figure-3 provides 40 keys. The keyboard has actually 8-rows and 5-columns. The first ten keys will represent numbers 0-9, next twenty-six keys represent the alphabets A to Z and remaining four keys represent the symbols. The circuit shows that rows are connected with P0 (P0.0 to P0.7) and columns are connected with P2 (P2.0 to P2.4). As high potential is applied to rows and columns of the keyboard, initially they remain high (+5v). The columns and rows make contact only when a key is pressed.

When a key is pressed, the key must be identified by its column and the row, and the intersection of the column and row must change from high to low. To detect a pressed key, the microcontroller grounds all rows by providing “0” to the Port 0, then it reads the columns. If the data read from the columns P2.4 to P2.0 = 11111, no key has been pressed and the process continues until a key press is detected. However, if one of the column bits has zero, this means that a key press has occurred. For example, if P2.4 to P2.0 = 11110, shows a key in column 1 has been pressed. After a key press is detected, the microcontroller will go through the process of identifying the key. Starting from the top row, the microcontroller grounds it by providing a low to row connected to control line 13 (P0.0) only; then it reads the columns. If the data read is all 1’s, no key in that row is activated and the process is moved to the next row. It grounds the next row, reads the columns, and checks for any zero. This process continues until the row is identified. After identification of the row in which the key has been pressed, the next task is to find out which column the pressed key belongs to.

Finally microcontroller gets the key pressed by matching the logic combination of row and column. Then the pressed key assigned code is displayed on the LCD module.

Programming the AT89C51

As the programmer for AT89C51 microcontroller is available now it is good to program it using any serial or parallel programmer. Using assembler (ASM51) one can easily assemble *.asm extension file to *.hex (machine code) which can be burnt into the microcontroller.

When the system is started initial message on LCD “PRESS ANY KEY!!!” will be displayed. As soon as the key is pressed on the keyboard, the display is cleared and key assigned code will be displayed on the LCD module.

8.4 Software Flowchart and Description

Software flowchart for the LCD Keyboard interface is shown below in Figure 8-5.

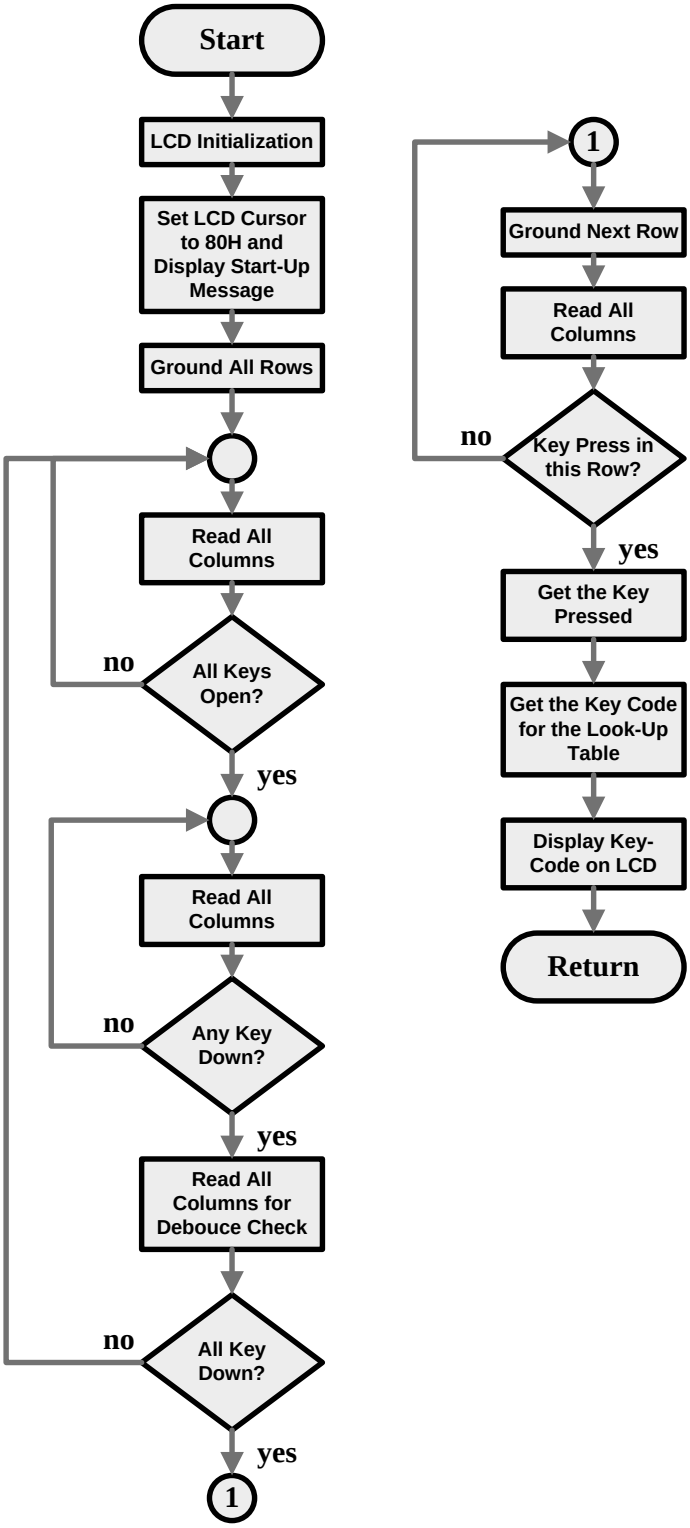


Figure 8-5: Software Flowchart for keyboard interface with AT89C51 Microcontroller.

The software is responsible for identifying the key pressed and displaying the key code on the LCD module. For this reading the codes and outputting the same to the proper Port at proper time is done through proper instructions of the microcontroller. In this case the software is written, such that the key press is displayed on the LCD module.

Software for keyboard interface and display initialization is self explanatory and is given as below:

```
; LCD & Keyboard Interface with Microcontroller
; AAB, 05:49 PM 16/11/2006
; Main program
$MOD51
        DB0         EQU         P1.0
        DB1         EQU         P1.1
        DB2         EQU         P1.2
        DB3         EQU         P1.3
        DB4         EQU         P1.4
        DB5         EQU         P1.5
        DB6         EQU         P1.6
        DB7         EQU         P1.7
        EN          EQU         P3.4
        RW          EQU         P3.3
        RS          EQU         P3.2
        DATA1       EQU         P1
              LJMP  MAIN
              ORG   150H
MAIN:
              LCALL     INIT_LCD        ; LCD initialization
              LCALL     CLEAR_LCD       ; Clear LCD
              CLR       RS
              MOV       DATA1, #80H     ; Set cursor at 80h
              SETB      EN
              LCALL     DELAY
              CLR       EN
              LCALL     WAIT_LCD
              MOV       A, #'P'
              LCALL     WRITE_TEXT      ; Write the message to LCD
              MOV       A, #'R'
              LCALL     WRITE_TEXT
              MOV       A, #'E'
              LCALL     WRITE_TEXT
              MOV       A, #'S'
              LCALL     WRITE_TEXT
              MOV       A, #'S'
              LCALL     WRITE_TEXT
              MOV       A, #' '
              LCALL     WRITE_TEXT
              MOV       A, #'A'
              LCALL     WRITE_TEXT
              MOV       A, #'N'
```

```
        LCALL   WRITE_TEXT
        NOP
        NOP
        NOP
        NOP
        NOP
        NOP
        NOP
        CLR     RS
        MOV     DATA1, #0C0H
        SETB    EN
        LCALL   DELAY
        CLR     EN
        LCALL   WAIT_LCD
        MOV     A, #'Y'
        LCALL   WRITE_TEXT
        MOV     A, #' '
        LCALL   WRITE_TEXT
        MOV     A, #'K'
        LCALL   WRITE_TEXT
        MOV     A, #'E'
        LCALL   WRITE_TEXT
        MOV     A, #'Y'
        LCALL   WRITE_TEXT
        MOV     A, #'!'
        LCALL   WRITE_TEXT
        MOV     A, #'!'
        LCALL   WRITE_TEXT
        MOV     A, #'!'
        LCALL   WRITE_TEXT
        MOV     P2, #0FFH       ; Make P2 input port
AAB1:   MOV     P0, #00H        ; Ground all rows
        MOV     A, P2           ; Read all columns
        ANL     A, #1FH         ; Mask unused bits
        CJNE    A, #1FH, AAB1   ; Check till all keys released
AAB2:   LCALL   DELAY           ; Call 20ms delay
        MOV     A, P2           ; See if any key is pressed
        ANL     A, #1FH         ; Mask unused bits
        CJNE    A, #1FH, OVER   ; Check for key press
        SJMP    AAB2            ; Jump to check again
OVER:   MOV     P0, #0FEH       ; Ground row 1
        MOV     A, P2           ; Read all columns
        ANL     A, #1FH         ; Mask unused bits
        CJNE    A, #1FH, ROW1   ; Key in row 1, find column
        MOV     P0, #0FDH       ; Ground row 2
        MOV     A, P2           ; Read all columns
        ANL     A, #1FH         ; Mask unused bits
        CJNE    A, #1FH, ROW2   ; Key in row 2, find column
        MOV     P0, #0FBH       ; Ground row 3
        MOV     A, P2           ; Read all columns
```

```
                ANL     A, #1FH          ; Mask unused bits
                CJNE    A, #1FH, ROW3    ; Key in row 3, find column
                MOV     P0, #0F7H        ; Ground row 4
                MOV     A, P2            ; Read all columns
                ANL     A, #1FH          ; Mask unused bits
                CJNE    A, #1FH, ROW4    ; Key in row 4, find column
                MOV     P0, #0EFH        ; Ground row 5
                MOV     A, P2            ; Read all columns
                ANL     A, #1FH          ; Mask unused bits
                CJNE    A, #1FH, ROW5    ; Key in row 5, find column
                MOV     P0, #0DFH        ; Ground row 6
                MOV     A, P2            ; Read all columns
                ANL     A, #1FH          ; Mask unused bits
                CJNE    A, #1FH, ROW6    ; Key in row 6, find column
                MOV     P0, #0BFH        ; Ground row 7
                MOV     A, P2            ; Read all columns
                ANL     A, #1FH          ; Mask unused bits
                CJNE    A, #1FH, ROW7    ; Key in row 7, find column
                MOV     P0, #07FH        ; Ground row 8
                MOV     A, P2            ; Read all columns
                ANL     A, #1FH          ; Mask unused bits
                CJNE    A, #1FH, ROW8    ; Key in row 8, find column
                LJMP    AAB2             ; Keep doing the same task
ROW1:           MOV     DPTR, #CODE1     ; Set Data pointer to code1
                LJMP    GET              ; Get the column and key code
ROW2:           MOV     DPTR, #CODE2     ; Set Data pointer to code2
                LJMP    GET              ; Get the column and key code
ROW3:           MOV     DPTR, #CODE3     ; Set Data pointer to code3
                LJMP    GET              ; Get the column and key code
ROW4:           MOV     DPTR, #CODE4     ; Set Data pointer to code4
                LJMP    GET              ; Get the column and key code
ROW5:           MOV     DPTR, #CODE5     ; Set Data pointer to code5
                LJMP    GET              ; Get the column and key code
ROW6:           MOV     DPTR, #CODE6     ; Set Data pointer to code6
                LJMP    GET              ; Get the column and key code
ROW7:           MOV     DPTR, #CODE7     ; Set Data pointer to code7
                LJMP    GET              ; Get the column and key code
ROW8:           MOV     DPTR, #CODE8     ; Set Data pointer to code8
GET:            RRC     A                ; Rotate Acc rt. once with carry
                JNC     GETDATA          ; Check for no carry
                INC     DPTR             ; Increment DPTR by one
                LJMP    GET              ; Back in loop
GETDATA:        CLR     A
                MOVC    A, @A+DPTR       ; Get the key code from look-
                MOV     R7, A            ; up table
                LCALL   CLEAR_LCD        ; Clear LCD
                MOV     A, R7
                LCALL   WRITE_TEXT       ; Write the key code on LCD
                LJMP    AAB1             ;Scan for the next key pressed
INIT_LCD:                                ; LCD initialization sub-routine
```

```
                CLR     RS
                MOV     DATA1, #38H
                SETB    EN
                LCALL   DELAY
                CLR     EN
                LCALL   WAIT_LCD
                CLR     RS
                MOV     DATA1, #0EH
                SETB    EN
                LCALL   DELAY
                CLR     EN
                LCALL   WAIT_LCD
                CLR     RS
                MOV     DATA1, #06H
                SETB    EN
                LCALL   DELAY
                CLR     EN
                LCALL   WAIT_LCD
                RET
CLEAR_LCD:                          ; Clear LCD sub-routine
                CLR     RS
                MOV     DATA1, #01H
                SETB    EN
                LCALL   DELAY
                CLR     EN
                LCALL   WAIT_LCD
                RET
WRITE_TEXT:                         ; Data write LCD sub-routine
                SETB    RS
                MOV     DATA1, A
                SETB    EN
                LCALL   DELAY
                CLR     EN
                LCALL   WAIT_LCD
                RET
WAIT_LCD:                           ; LCD delay sub-routine
                MOV     R2, #03H
                MOV     R3, #0FFH
                CLR     EN
                CLR     RS
                SETB    RW
                MOV     DATA1, #0FFH
                SETB    EN
                MOV     A, DATA1
LOOP:           JB      ACC.7, HERE2
HERE2:          NOP
                NOP
                DJNZ    R3, HERE2
                DJNZ    R2, LOOP
                CLR     EN
```

```
            CLR     RW
            LCALL   DELAY
            RET
DELAY:                          ; Delay sub-routine for keyboard
            MOV     R2, #37
AGAIN:      MOV     R3, #255
HERE1:      NOP
            NOP
            DJNZ    R3, HERE1
            DJNZ    R2, AGAIN
            RET
            ORG     900H        ; Look-up table for key code
CODE1:      DB      '1', '3', '5', '7', '9'
CODE2:      DB      '2', '4', '6', '8', '0'
CODE3:      DB      'A', 'C', 'E', 'G', 'I'
CODE4:      DB      'B', 'D', 'F', 'H', 'J'
CODE5:      DB      'K', 'M', 'O', 'Q', 'S'
CODE8:      DB      'L', 'N', 'P', 'R', 'T'
CODE7:      DB      'U', 'W', 'Y', '?', '&'
CODE6:      DB      'V', 'X', 'Z', '.', '!'
            END
```

Software Explanation

Software provides LCD initialization and data write function for the LCD module. LCD is cleared and message “PRESS ANY KEY!!!” is displayed on the LCD module. After that the program for detection and identification of key activation is started. P0 and P2 are initialized as output and input, respectively. The major stages of the program are listed below:

- For all keys have been released, “0s” are output to all rows (P0) at once, and the columns (P2) are read and checked repeatedly until all the columns are high. When all columns are found to be high, the program waits for a short amount of time before it goes to the next stage of waiting for a key to be pressed.
- For any key closure, the columns are scanned over and over in an infinite loop until one of them has a “0” on it. After the key press detection, it scans the columns (P2) again. It ensures that the first key press detection was not an erroneous one. It goes to the next stage to detect which row it belongs to; otherwise, it goes back into the loop to detect a real key press.
- For which row the key belongs to, it grounds one row at a time by sending appropriate code to Port 0 and reading the columns each time. If it finds that all columns are high, this means that the key press cannot belong to that row; therefore, it grounds the next row and continues until it finds the row the key press belongs to. After finding the row, it sets-up the starting address for the look-up table holding the key codes for that row and goes to the next stage to identify the key.
- For any key press, it rotates the columns bits right, one bit at a time, in the carry flag and checks to see if it is low(“0”). If the carry flag is set

DPTR is incremented once. Upon finding the zero, it pulls out the ASCII code for that key from the look-up table.

- When the key code identified and transferred to accumulator from look-up table, the display is cleared and key code is written on the LCD module. The program is looped back for the other key press detection.

Component layout diagram

Figure 8-6 shows the component side of PCB for the keyboard interface with AT89C51 microcontroller.

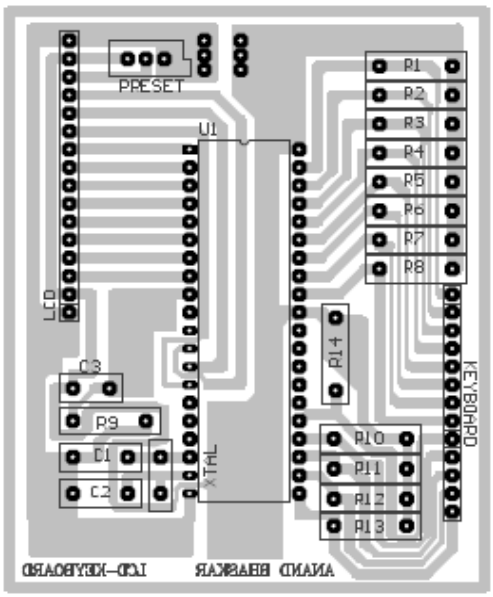


Figure 8-6: Actual size component side PCB for the keyboard interface with AT89C51 microcontroller.

Part List

Semiconductors:

U1	- AT89C51 or its derivative 8-bit microcontroller
U2	- 1 x 16 LCD Modules
U3	- 10 x 4 Matrix Keyboard

Resistor:

R1-R14	- 10k ohms
R9	- 8.2k ohms
POT	- 10k ohms

Capacitors:

C1, C2	- 27pf
C3	- 10uf

Miscellaneous:

XTAl	- 11.0592MHz

Wire, 5V power supply, Programmer to program 8-bit microcontroller, Assembler ASM51 or any to assemble *.asm file to get error free *.lst and *.hex extention file.

Solder Side PCB for LCD-Keyboard Interface

Figure 8-7 shows the actual size single side PCB for the keyboard interface with AT89C51 microcontroller.

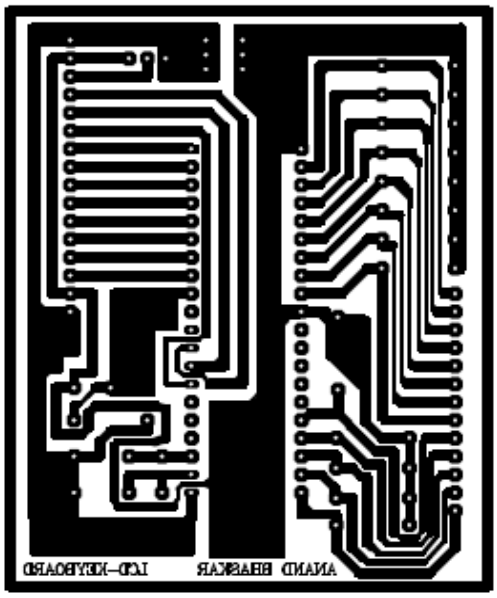


Figure 8-7: Actual size single side PCB for the keyboard interface with AT89C51 microcontroller.

8.5 Future Developments

There are many applications where LCD-Keyboard interface can fit. Many of the application require the commands to be given by the keyboard and proper message on the LCD shows the working of the system as per the displayed message. Universal LCD-Keyboard terminal can be made which can fit into any application like Vending machine auto programming, Keypad lock system, Industrial automation and control applications etc.

Chapter 9 DOT-Matrix Display Interface: Moving Message Display

A moving message display system is discussed in this chapter. The system includes AT89C51 microcontroller and three 5x7 LED matrix. The designing, circuit explanation and software development is presented.

1.1 Introduction

Moving message displays are now days getting more importance for better advertisement. Due to different varieties and versatility such displays are useful at various places, for example, to show the information for the trains in railway stations or in hotel lounges to assist the hosts. These displays are available starting from simple LED's to LCD modules.

It has been subject of interest for the hobbyist and the students of electronics field to know how such displays may be working. We are presenting here a simple way to design and construct a moving message display system using AT89C51 microcontroller.

1.2 About Display Module

The present work is centered on the interfacing of microcontroller AT89C51 and display modules. We have considered here the 5x7 matrix modules made of Kwality Photonics Private Limited, the module number is KLP1057I. It is a common anode type of module. The details of this module are shown in Figure 9-1 and Figure 9-2.

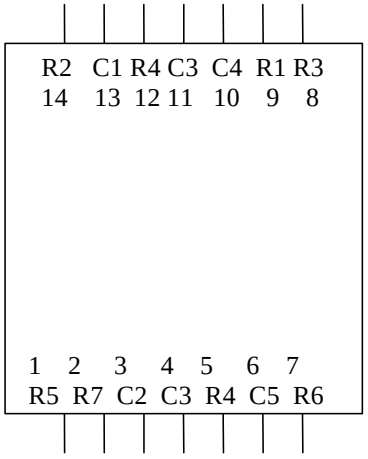


Figure 9-1: Pin configuration of 5x7.

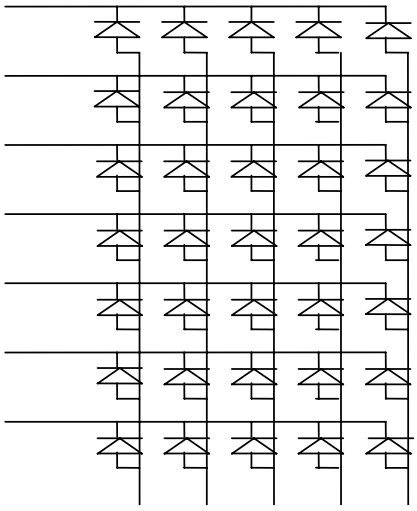


Figure 2: Internal diagram of 5x7 display module.

Figure 1, describes the pin arrangement of seven rows and five columns. Figure 2 shows the internal arrangement of LED's in 5x7 matrix. One can note that in order to lit any particular LED one has to connect the

positive voltage to the corresponding column and ground the corresponding row.

In our circuit we have combined three such modules resulting in a 15x7 matrix display. Thus AT89C51 microcontroller controls the seven rows and fifteen columns of the display. The corresponding rows and columns of the display are activated under the control of software written for microcontroller to create the effect of moving characters.

9.3 Detailed Circuit Diagram and Explanation

The circuit diagram for the moving message display is shown in Figure 9-3.

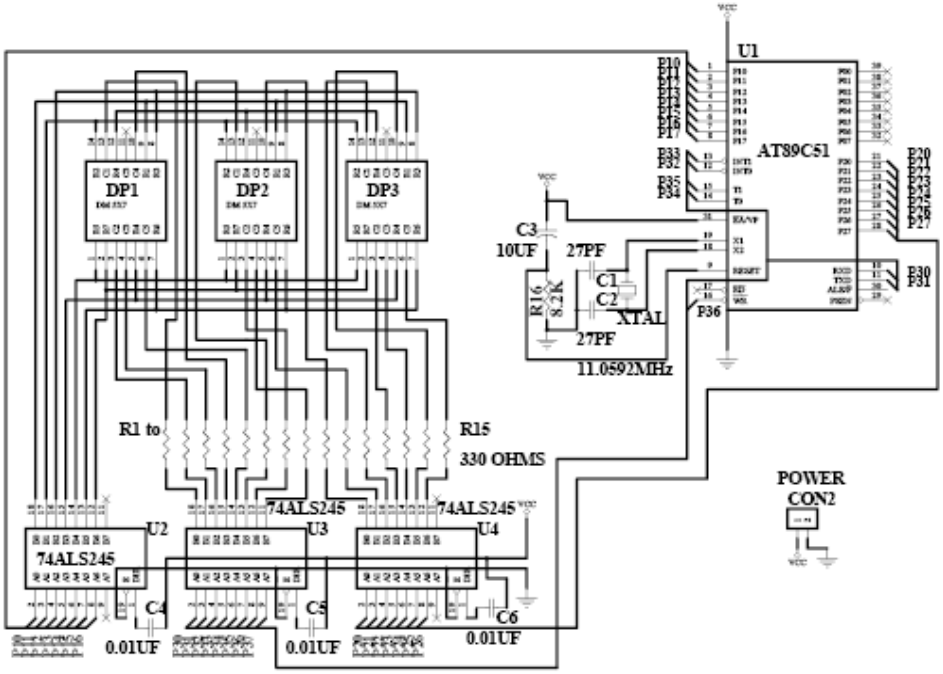


Figure 9-3: Circuit diagram of moving message display.

We have connected the corresponding rows of all the three displays in series. Hence there are seven rows and fifteen columns in the display. This can be visualized from Figure 9-4.

All the seven rows of the display are connected to inputs of 74LS245 that is IC U2. The output of the same IC is connected to port1 of microcontroller. Out of total fifteen columns, i.e. from C1 to C15, columns C1 to C7 are connected to port2 of microcontroller through 74LS245 IC U4. Similarly the remaining eight columns C8 to C15 are connected to port3 of microcontroller through 74LS245 IC U3. The microcontroller is provided with power on reset and a working frequency of 11.0592 MHz.

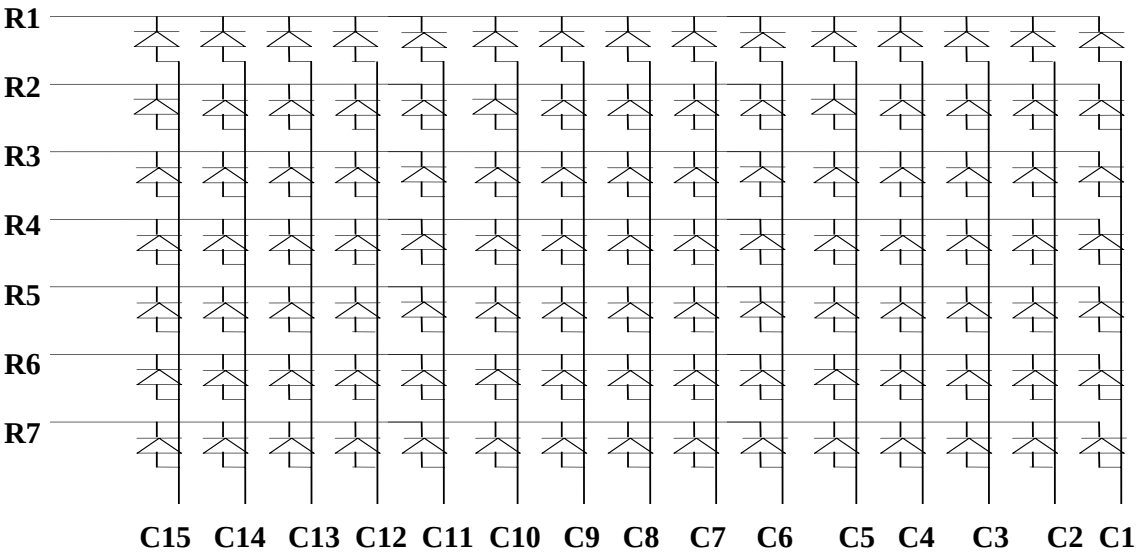


Figure 9-4: Internal diagram of 15x7 display module.

The connection details of modules with the microcontroller ports are shown in detail in Table 9-1.

Signal from display	AT89c51 Port Pins
DP1-DP2-DP3	
R1-U2-A0	P1.0
R2-U2-A1	P1.1
R3-U2-A2	P1.2
R4-U2-A3	P1.3
R5-U2-A4	P1.4
R6-U2-A5	P1.5
R7-U2-A6	P1.6
DP1	
C1-U3-A0	P3.0
C2-U3-A1	P3.1
C3-U3-A2	P3.2
C4-U3-A3	P3.3
C5-U3-A4	P3.4
DP2	
C1-U3-A5	P3.5
C2-U3-A6	P3.6
C3-U3-A7	P3.7
C4-U4-A0	P2.0
C5-U4-A1	P2.1
DP3	
C1-U4-A2	P2.2
C2-U4-A3	P2.3
C3-U4-A4	P2.4
C4-U4-A5	P2.5
C5-U4-A6	P2.6

Table 9-1: Connection Details of Display Module Connected to AT89c51 Microcontroller Ports.

Circuit Function

To display character pattern, certain data codes must be applied to the buffer IC 74LS245 inputs A0 to A7 whose outputs B0 to B7 are connected to 15x7 matrix display by means of microcontroller port output. Port1 is used to control the seven rows. Port3 and Port2 are used to control the fifteen columns of 15x7 matrix display. Figure 9-5, shows the block diagram of the moving message display.

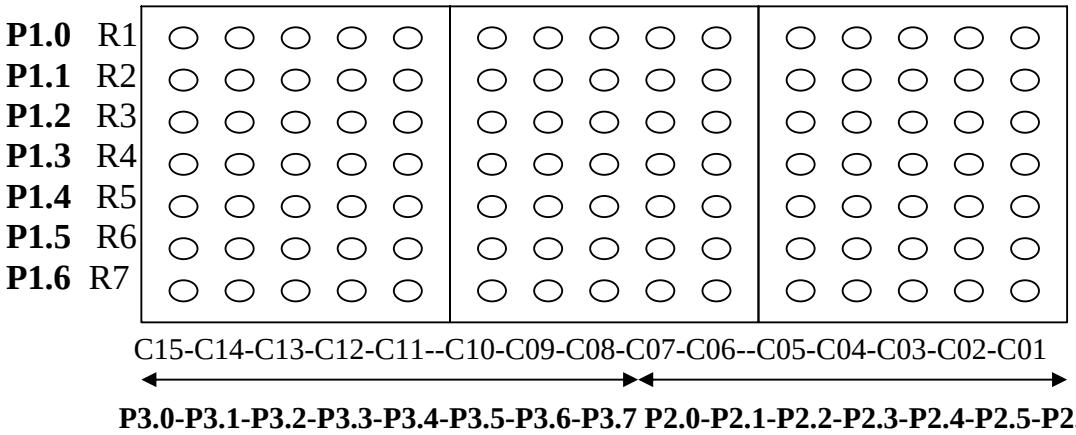


Figure 9-5: Correspondence of ports with Row-Column of 15x7 display.

Each column can be selected by sending corresponding data to Port3 and Port2. For example to select the first column (C01) one has to send 00h to Port3 and 40h to Port2. Table 9-2 shows the column selection code from C01 to C15.

COLUMN	PORT3								PORT2							
	P 3.7	**P 3.6**	**P 3.5**	**P 3.4**	**P 3.3**	**P 3.2**	**P 3.1**	**P 3.0**	**P 2.7**	**P 2.6**	**P 2.5**	**P 2.4**	**P 2.3**	**P 2.2**	**P 2.1**	**P 2.0**
C01 (0040H)	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0
C02 (0020H)	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0
C03 (0010H)	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0
C04 (0008H)	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0
C05 (0004H)	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0
C06 (0002H)	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0
C07 (0001H)	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
C08 (8000H)	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
C09 (4000H)	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0
C10 (2000H)	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0
C11 (1000H)	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0
C12 (0800H)	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0
C13 (0400H)	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0
C14 (0200H)	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0
C15 (0100H)	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0

Table 9-2: Column selection codes for 15x7 matrix moving message display.

Note that P2.7 is not connected so treated as “0” in code generation.

When single column is selected one has to provide the corresponding row selection codes also, to lit the LED’s of selected column. The row selection code differs as per the user requirement for the message to display. To display “5x7” on the module one has to select the column first and provide row the data codes. Each row-column selection pattern is separated by less than 1s delay. This switching delay between the row-column selection pattern is such that human eye can not detect the single row-column selection pattern and form a continuous character pattern of “5x7” in this case. Table 9-3 shows the row selection codes to be provided when particular column is selected to display message “5x7”.

COLUMN SELECTED	ROW CODE	ROWS R7 TO R1 - PORT1							
		P1.7 (NC)	P1.6 (R7)	P1.5 R(6)	P1.4 (R5)	P1.3 (R4)	P1.2 (R3)	P1.1 (R2)	P1.0 (R1)
C15	30H	0	0	1	1	0	0	0	0
C14	36H	0	0	1	1	0	1	1	0
C13	36H	0	0	1	1	0	1	1	0
C12	36H	0	0	1	1	0	1	1	0
C11	0EH	0	0	0	0	1	1	1	0
C10	3BH	0	0	1	1	1	0	1	1
C09	57H	0	1	0	1	0	1	1	1
C08	6FH	0	1	1	0	1	1	1	1
C07	57H	0	1	0	1	0	1	1	1
C06	3BH	0	0	1	1	1	0	1	1
C05	3EH	0	0	1	1	1	1	1	0
C04	5EH	0	1	0	1	1	1	1	0
C03	6EH	0	1	1	0	1	1	1	0
C02	76H	0	1	1	1	0	1	1	0
C01	78H	0	1	1	1	1	0	0	0

Table 9-3: Row-Column selection code to generate character pattern of “5x7”.

By means of software burnt in microcontroller these codes are output to three ports and corresponding pattern is created on display module. This can be visualized in Figure 9-6.

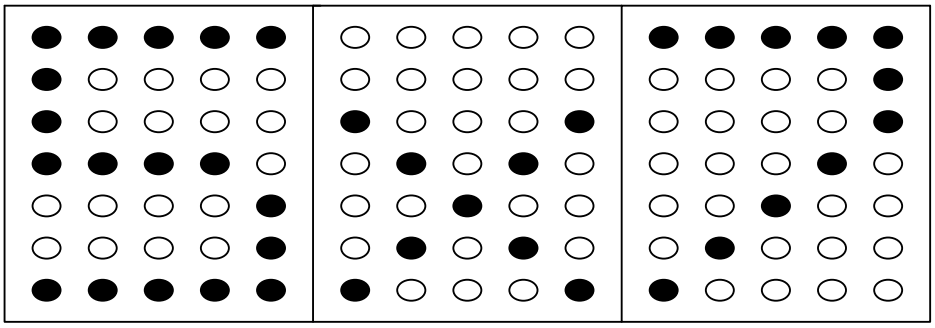


Figure 9-6: “5x7” message pattern on display module.

By means of software appropriate column is selected and row codes for each digit (i.e. 5-width 7-hieght) are fetched from the ROM. Row codes are saved on memory location 700h onwards. For moving the message “5x7 HNP”, shift the row data selection right side once as the data on ROM are 700h onwards. So for first shift DPTR (data pointer) must be set to 701h and for continuous shifting increment the DPTR till the last data saved. For better understanding, Figure 9-7 shows the step by step display pattern of the module.

After the completion of last step, data 7fh is given to the port 1 twelve more times selecting each time the corresponding column so that the display will be cleared and again loop back to step 1. In this way moving message “5x7 HNP” is displayed.

Programming the AT89c51

Any flash microcontroller can be used for this application. As the programmer for AT89c51 is available now it is good to program it using any serial or parallel programmer. Using assembler (ASM51) one can easily change *.asm extension file to *.hex (machine codes) which can be burnt in to the microcontroller.

Program given here will display moving message of “5x7 HNP” continuously.

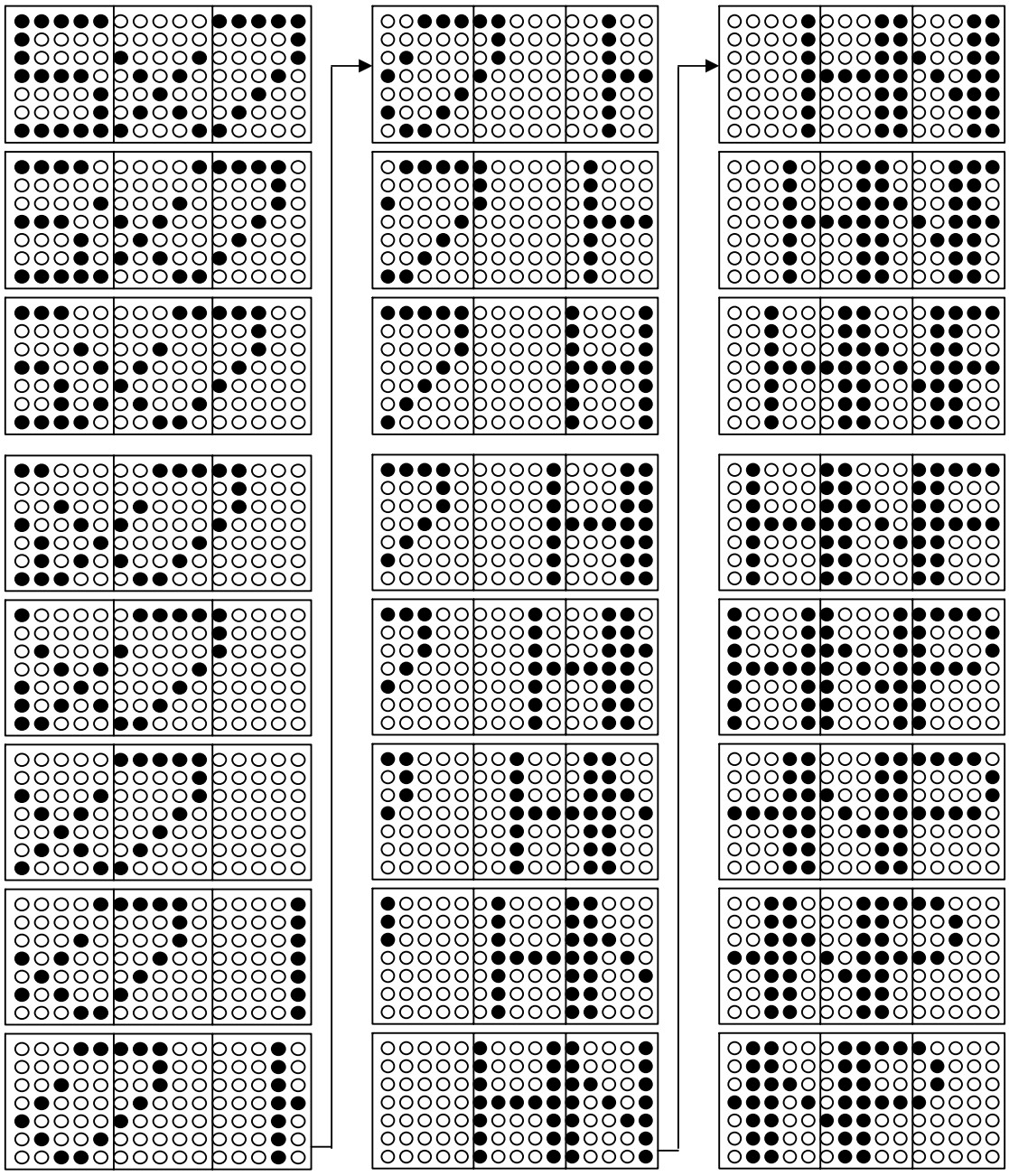


Figure 9-7: Step by step display pattern of the module.

9.4 Software Flowchart and Description

Figure 9-8 shows the flowchart of the software used for moving message display.

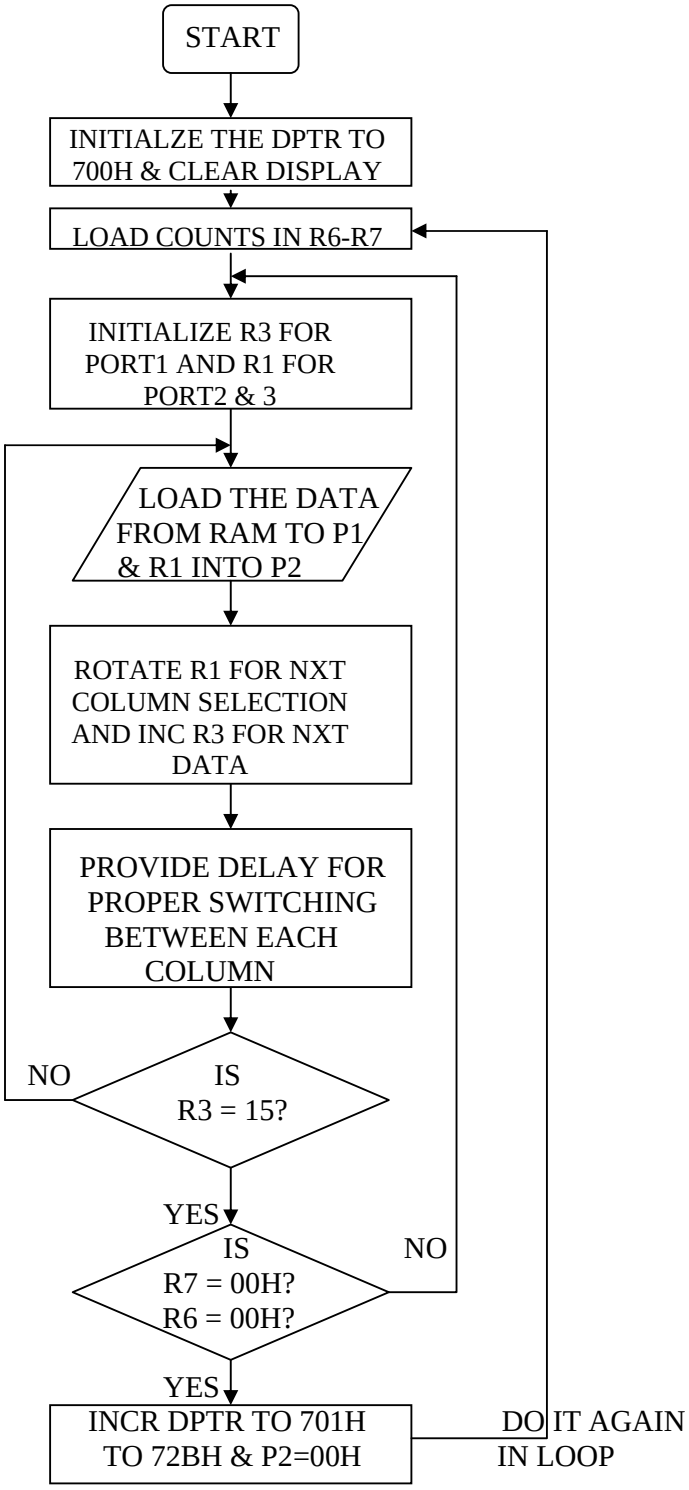


Figure 9-8: Software flowchart.

The software is responsible to maintain the movement of the characters on the display. For this reading the codes and outputting the same to the proper port at proper time is done through proper instructions of the microcontroller. In the present case the software is written, such that the message '5x7 HNP' is displayed sequentially.

Software for display initialization and data write is self explanatory and is given as below:

```
; Moving message display using 15x7 matrix
; Main program
$MOD51
                LJMP    MAIN
                ORG     150H
MAIN:   MOV     SP, #40H        ; Move stack pointer to 40H
NEXT:   MOV     DPTR, #700H     ; Load Data pointer = 700H
        LCALL   CLRDISP         ; Clear display
        LCALL   DATANXT         ; Call data and display to module
        MOV     R0, #24H        ; Store no. of data in R0
NEXT1:  INC     DPTR            ; Increment DPTR to point next data
        MOV     P2, #00H        ; Clear port2
        LCALL   DATANXT         ; Call data and display to module
        DJNZ    R0, NEXT1       ; If R0 is not zero jump to next1
        LJMP    NEXT            ; Run program in continuous loop
DATANXT:
        MOV     R6, #01         ; Initialize count in R6
AB2:    MOV     R7, #100        ; Initialize count in R7
AB1:    MOV     R3, #00H        ; R3 = 00h for 1st code from memory
        MOV     R1, #01H        ; Column 1 selection of P3
NXTDATA:
        MOV     A, R3           ; Get the data from memory
        MOVC    A,@A+DPTR
        MOV     P1, A           ; Sent data to P1 for row codes
        MOV     P2, #00H        ; Off port2 connected display

        MOV     P3, R1          ; Select column of display
        MOV     A, R1           ; Get the selection code of
        RL      A               ; column in R1 again
        MOV     R1, A
        INC     R3              ; point next data on memory
        LCALL   DELAY1          ; provide switching delay
        CJNE    R3, #08, NXTDATA; R3<=8 get the next data
        MOV     R1, #01H        ; Column 1 selection of P2
NXTDATA1:
        MOV     A, R3           ; Get the data from memory
        MOVC    A,@A+DPTR
        MOV     P1, A           ; Sent data to P1 for row codes
        MOV     P3, #00H        ; Off port3 connected display
        MOV     P2, R1          ; Select column of display
```

```
        MOV     A, R1           ; Get the selection code of
        RL      A               ; column in R1 again
        MOV     R1, A
        INC     R3              ; point next data on memory
        LCALL   DELAY1          ; provide switching delay
        CJNE    R3, #15, NXTDATA1; R3<=15 get the next data
        DJNZ    R7, AB1         ; If R7#00h jump to AB1
        DJNZ    R6, AB2         ; If R6#00h jump to AB2
        RET
CLRDISP:
        MOV     P1, #0FFH       ; Clear the display
        MOV     P2, #00H
        MOV     P3, #00H
        RET
DELAY1:
        MOV     R5, #01         ; Switching delay between columns
HERE2:  MOV     R4, #230
HERE1:  DJNZ    R4, HERE1
        DJNZ    R5, HERE2
        RET
        ORG     700H            ; Codes for P1
VALUE:  DB      30H, 36H, 36H, 36H, 0EH, 3BH, 57H, 6FH
        DB      57H, 3BH, 3EH, 5EH, 6EH, 76H, 78H, 7FH
        DB      7FH, 7FH, 7FH, 7FH, 00H, 77H, 77H, 77H
        DB      00H, 00H, 7BH, 77H, 6FH, 00H, 00H, 76H
        DB      76H, 76H, 79H, 7FH, 7FH, 7FH, 7FH, 7FH
        DB      7FH, 7FH, 7FH, 7FH, 7FH, 7FH, 7FH, 7FH
        DB      7FH, 7FH, 7FH
        END
```

Software Explanation

Software provides initialization and data write function for the display module. Row selection codes are controlled by Port 1 of microcontroller and are stored in memory location at address 700h. These row selection codes can be fetched by means of data pointer (DPTR). Initially data pointer is set to 700h memory location which points to the first row selection data code. The display is cleared by providing the opposite logic to Port 1, Port 2, and Port 3 by means of the sub-routine CLRDISP. For the data write on display module DATANXT sub-routine is executed in the program. In DATANXT sub-routine R1 is used for column selection. R3 set to 00h to get first code from data pointer address (700h). R6 and R7 provides the continuous display for one whole fifteen data write function for certain period of time (approx. 1 sec.). Initially the value of R3 = 00h so when first time the instruction MOVC A,@a+DPTR is executed the data present at memory location 700h is stored in A. That is the data for row selection is sent to Port 1. Now the column is selected to display the row data. So the register R1 is used to provide the P3 the column selection codes. After that rotating the data of R1 by means of accumulator using RLA instruction to get the next column selection code. Incrementing DPTR provides next row selection codes. As the Port 3 eight bits are controlling the eight column selection codes we are comparing the R3 with 08h by means of

CJNE R3,#08,NXTDATA. If R3 is less than 08h the program jumps to NXTDATA otherwise steps down and executes the next instruction. After eight data write column selection, Port 2 is activated to control the next seven column selection codes. NXTDATA1 is same as NXTDATA, the only difference is in column selection which is now controlled by Port 2. After the completion of DATANXT routine DPTR is incremented once and column selection codes are left as it is, so that the display moved the one column left. And fetch the new column selection (C01) on very right side of the display module. Again the DATANXT is called to provide the data write function to the microcontroller ports. This increment in DPTR causes the program to start fetching row codes from next memory address 700h onwards which in result visualizes the moving message on the display.

In this program appropriate delay is provided by means of DELAY1 sub-routine so that change in each row-column selection code is not visualized by human eye and constant moving message is displayed on the 15x7 display module.

Component layout diagram

Figure 9-9 shows the component side of PCB for the moving message display.

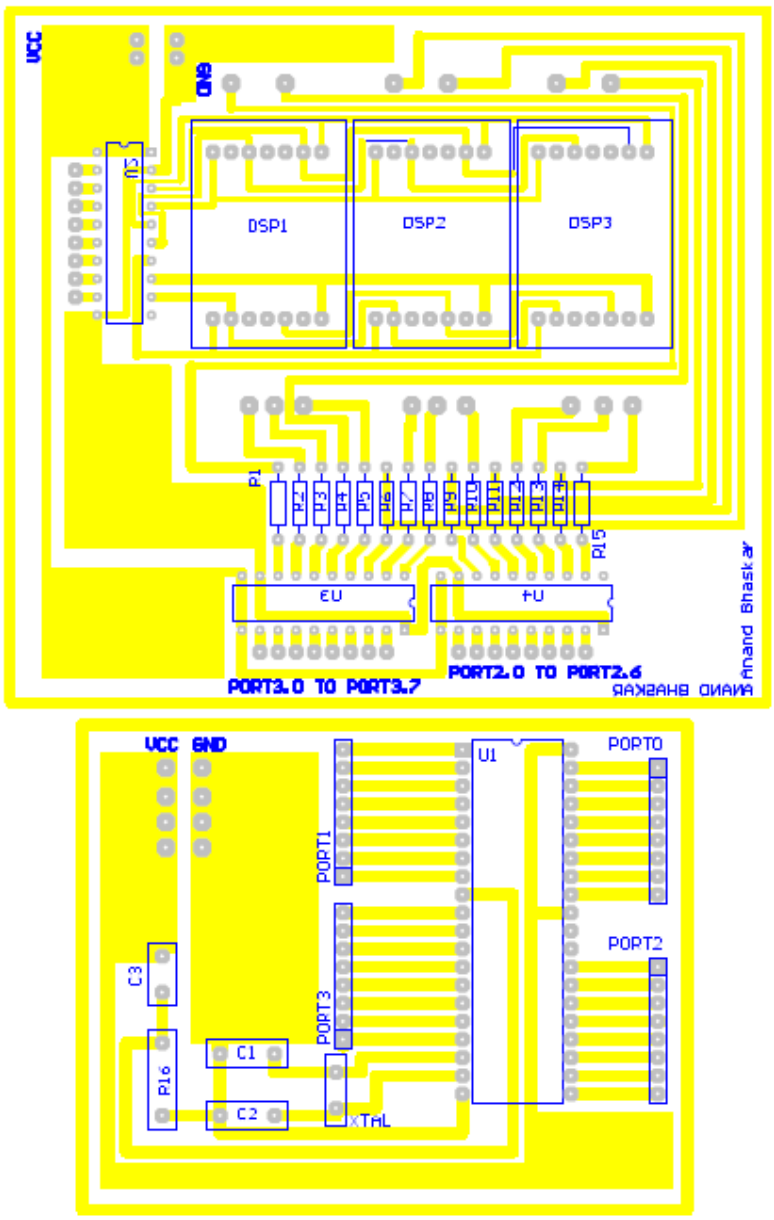


Figure 9-9: Actual size component side PCB for the moving message display.

Solder Side PCB for Moving Message Display

Figure 9-10 shows the actual size single side PCB for the moving message display.

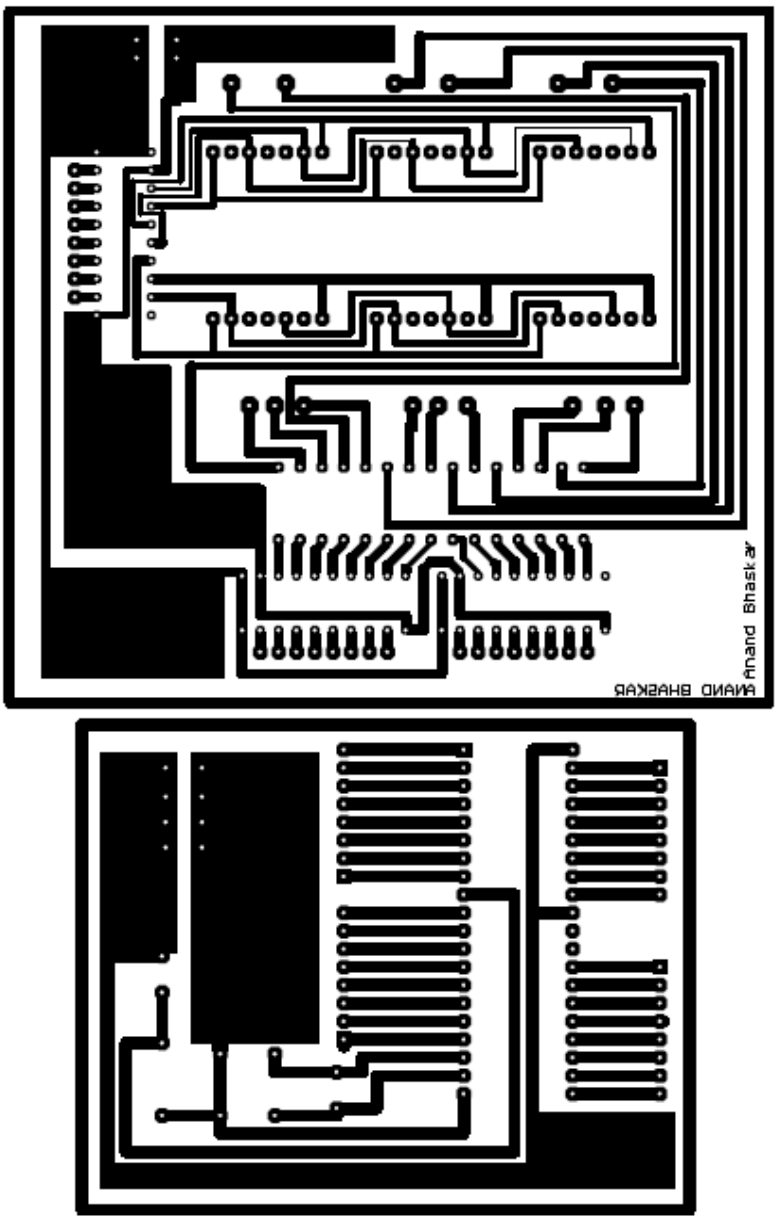


Figure 9-10: Actual size single side PCB for the moving message display

9-5 Future Developments

Both common anode and cathode type displays are available in the market. They can be used as per logic of the design. For future perspective this sort of moving message displays can be interfaced with the computer so the desired message can be displayed at any time as per the user requirement. One of the problems with the design of such kind of moving message displays is the current sourced by individual led's in the dot matrix, better algorithms can be developed which can help the designer to glow message led's more brighter and with less blurring.

Chapter 10 RS232 Interface: Serial Communicator

In this chapter a simple communication system between AT89C51 and personal computer is discussed. The system includes AT89C51 microcontroller, LED panel & MAX-232. The detail circuit diagram and development of hardware and software is presented here.

1.1 Introduction

The most basic, simple and intelligent communication interface of the microcontroller with another active system is the Serial Communication, called RS-232 by electronics technicians. RS-232 has been the popular standard for inter-computer communication since 1950s before it was standardized by Electronic Industries Association (EIA).

Although PC-to-PC communication is possible with three wires only, we are here concentrating on PC-to-AT89C51 microcontroller communication and its basics. We also glance on the .NET programming paradigm, which makes the system possible to develop faster, easily and secure; without going into details of port address of serial COM.

1.2 Basic Understanding

The name itself says about the type of communication being processed, i.e. the data is transferred as well as received serially. The common and simple asynchronous communication between PC and microcontroller is supported by RS-232. AT89C51 possesses an internal peripheral serial port, which works on standard of RS-232 that makes the world of microcontroller technicians easy. This method had lead to many new applications including, programming the Flash of the microcontroller serially with an ease. Now we discuss the concept behind the popular serial interface used here.

The popularity of a communication interface fully depends upon these three properties:

1. Message: The information itself.
2. Message format: The predefined pattern of structuring the message.
3. Protocols: A set of rules with which the computers will communicate.

Comparison of popular common computer communication interfaces with microcontrollers is shown in Table 10-1.

In serial data transmission and reception, data is transmitted 1 bit at a time and vice-versa. This format is only possible with the use of clocking the data in and out with the help of their internal clocks. Based on the type of clocks used there are two serial-data formats: Synchronous transmission and Asynchronous transmission.

Interface Name	Modes	Device connection possible	Length (max-feet)	Speed (max) Bit/sec
RS-232 EIA-TIA-232	**Asynchronous Serial**	**2**	**50-100**	**20k**
SPI	Synchronous Serial	8	10	2.1M
I^2C	Synchronous Serial	40	18	400K
Microwire	Synchronous Serial	8	10	2M
USB	Asynchronous Serial	127	16	12M
Parallel Printer Port	Parallel	2 or 8 with Daisy-chain support	10-30	1M

Table 10-1: Common Computer Communication Interfaces with Microcontroller.

RS-232 has different logic levels to that of microcontrollers as shown in Figure 10-1. AT89C51 is both CMOS and TTL logic compatible. RS-232 logic has to be converted to TTL for them to work mutually. This is possible by using an external interface chip MAX-232 from Maxim.

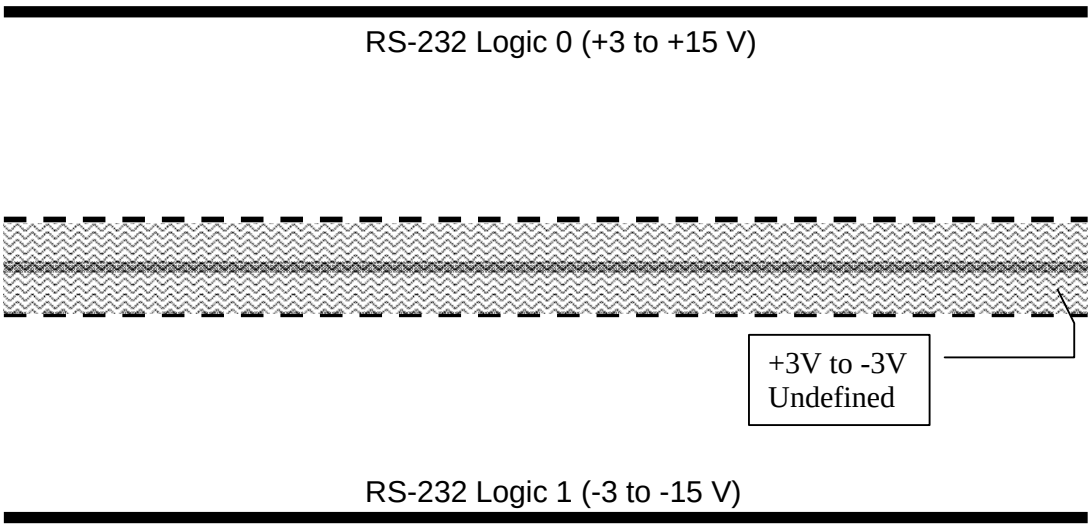


Figure 10-1: Logic Levels of RS-232

MAX-232 has two drivers that convert TTL inputs to RS-232 logic and vice-versa. There are 9 signals of concerned with RS-232, but we are using only three of them:

1. TD (Pin 3):-This transmits the data
2. RD (Pin 2):- This receives the data.
3. GND (Pin 5):- This is signal ground.

The diagrams of pin outs for RS-232 9 pin connectors are shown in Figure 10-2.

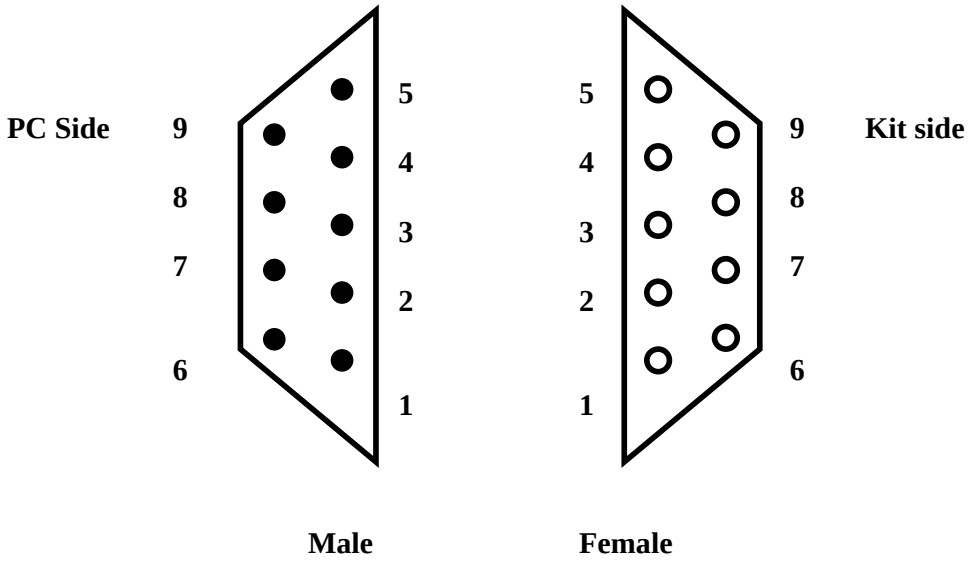


Figure 10-2: Pin outs for Rs-232 9 Pin Connectors

The baud rate, parity bit and stop bit must be defined and decided before the actual communication takes place. The most common format is 8-N-1. Start bit is send first, then the LSB of data, parity bit if required (not used here), and one stop bit. In this sequence each byte of data is send to the receiver. This is called message format and use of parity and stop bit is called protocols that is understood between the microcontroller and PC.

10.3 Detailed Circuit Diagram and Explanation

Figure 10-3, shows the schematic diagram of microcontroller serial communicator simplified.

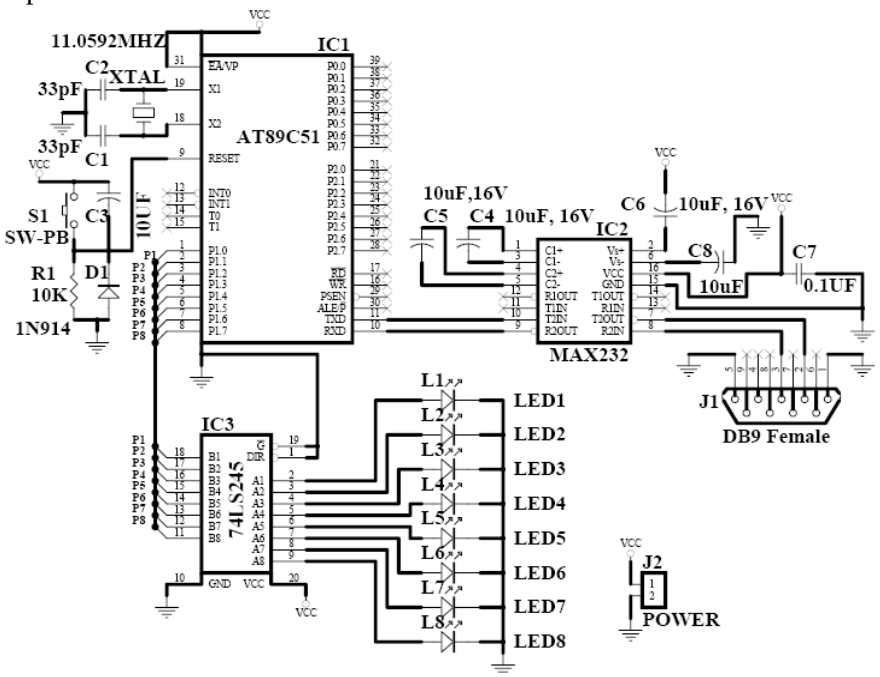


Figure 10-3: Schematic Diagram of Microcontroller Serial Communicator.

The circuit diagram includes AT89C51 microcontroller IC1, serial interface IC2 MAX-232 and IC3 74LS245 bi-directional buffer. Here a crystal of 11.0592 MHz (XTAL), that clocks the AT89C51, is used so that a standard communication baud rate of 9600. 74LS245 is used as unidirectional non inverting buffer. MAX-232 converts the logic between RS-232 and TTL. Eight Light emitting diodes from L1 to L8 are placed to determine the ASCII values of a-z and numerals 0-9. Female serial DB-9 pin connecter is used to connect to the PC. A standard +5V supply is need to power-on the circuit. A push to ON reset switch is also placed to reset the circuit whenever needed.

Port 1 (pins 1-8) drives the output of the system. The output i.e. LEDs are driven by the 74LS245 buffer IC2. Pin 1 and Pin 19 are grounded to enable and set the direction of the data to be passed through it. Reset switch with capacitor C3 is used to reset the AT89C51.

Circuit Function

Figure 10-4 shows the circuit functional block diagram of microcontroller serial communicator simplified.

Block Diagram of
Microcontroller Serial Communicator Simplified

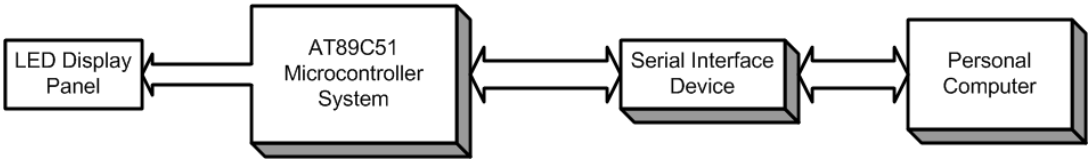


Figure 10-4: Block Diagram.

Pin 11 of At89C51 is TXD pin that transmits the data stored or processed inside the microcontroller to pin 10(T2IN) of MAX-232. MAX-232 then converts the TTL logic to RS-232 compatible and outputs it on Pin 7(T2OUT) which is connected to the Pin 3 of DB-9M COM port via female connector as shown in Figure 10-3. Data transmitted serially is than received at the COM port of PC and displayed in the PC screen.

Data sent by user by pressing any input keystroke (0-9, A-Z) is sent to RS232 DB-9M COM port Pin 2 which is connected to Pin 8 (R2IN) of IC2. MAX-232 converts the RS232 logic level to TTL and fed it to Pin 9(R2OUT). Pin 10 of AT89C51 is RXD pin that receives the data from pin 9 of MAX-232. Data received serially is then transferred to P1 of AT89C51 to display bits 0-9 or ASCII values of A-Z input key pressed by the user.
This is only the body of the project but the heart is the software program that runs within the hardware we discussed so for. We had developed microcontroller software in assembly language and front end for the PC for easy user interface in Visual C#.NET.

Programming the AT89C51

Any flash microcontroller can be used for this application. As the programmer for AT89c51 is available now it is good to program it using any serial or parallel programmer. Using assembler (ASM51) one can easily change *.asm extension file to *.hex (machine codes) which can be burnt in to the microcontroller. Using Visual Studio-2005 and MSDN library help one can easily develop the given front end.

The two way communication is implemented by sending data to PC and by receiving data from PC to microcontroller serially.

10.4 Software Flowchart and Description

Figure 10-5; shows the flowchart of the software used for microcontroller serial communicator simplified.

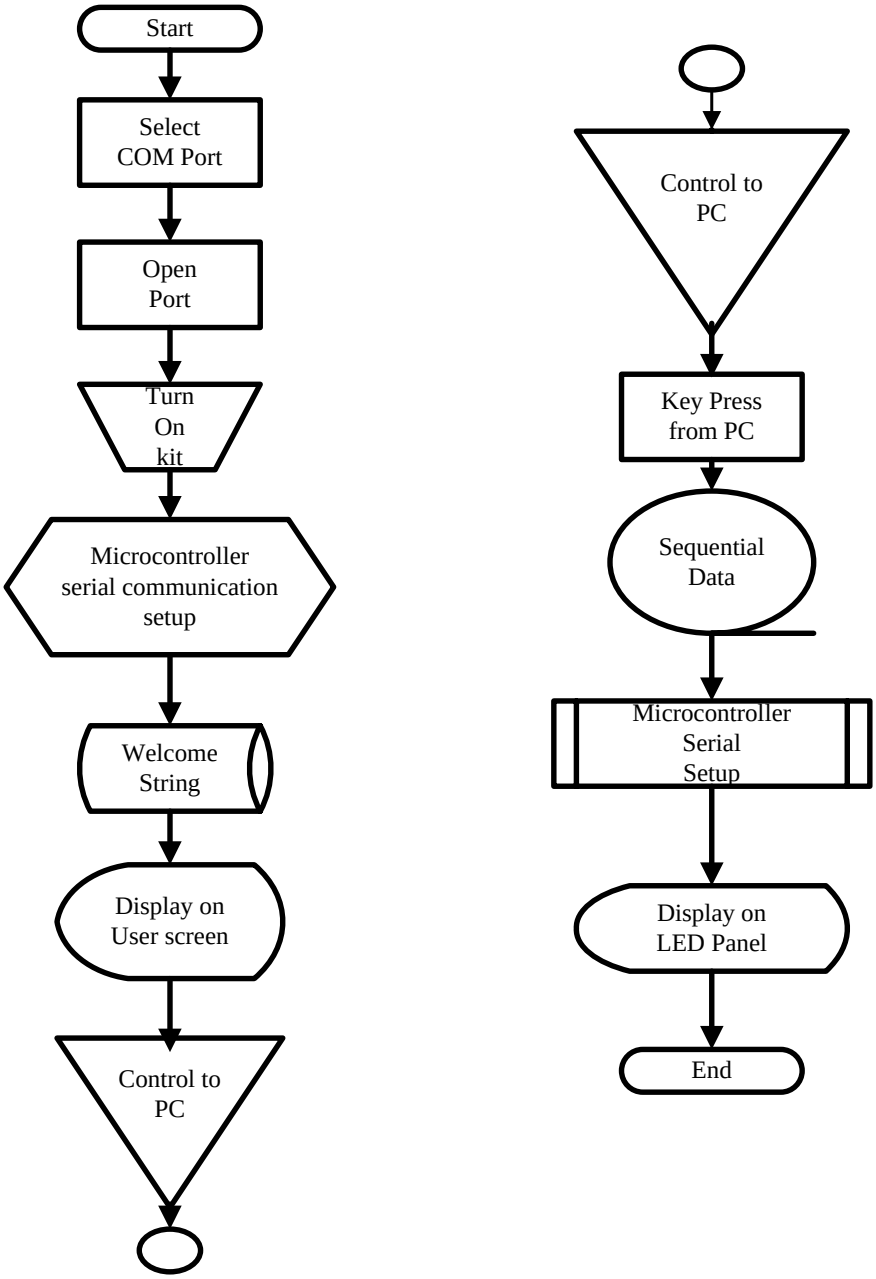


Figure 10-5: Software Flowchart.

Software Explanation

- **Assembly language software for microcontroller**

Software for data transfer and data receive is self explanatory and is given as below:

```
; Microcontroller Serial Communicator Simplified
; Program transfers the message stored at data field 200h to PC &
; Get ASCII character from PC and display it on LED's connected to P1.
; 04/02/2007
; Main program
$MOD51
                ORG  00H
                LJMP MAIN
                ORG  100H
MAIN:     MOV   TMOD, #20H      ; timer 1, mode 2 (auto-reload)
          MOV   TH1, #0FDH      ; 9600 baud rate
          MOV   SCON, #50H      ; 8-bit, 1 stop, REN enabled
          SETB  TR1             ; start timer 1
START:    MOV   DPTR, #200H     ; load pointer for message
AAB1:     CLR   A
          MOVC A,@A+DPTR        ; get the character
          LCALL TRANS           ; transmit it to PC com-port
          INC   DPTR            ; point to the next character
          JZ    HERE            ; if null then stop transmitting
          LJMP  AAB1            ; keep on transmitting
HERE:     JNB   RI, HERE        ; wait here for receive data from PC
          MOV   A, SBUF         ; save it in accumulator
          CLR   RI              ; get ready for next character
          CJNE  A, #30H, NT_EQUAL; recognize nos. and characters
          SUBB A, #30H          ; if in between 0-9 than remove ASCII bias
          LJMP  DISP            ; display it to LED's, port P1
NT_EQUAL: CJNE  A, #31H, NT_EQUAL1
          SUBB A, #30H
          LJMP  DISP
NT_EQUAL1:CJNE A, #32H,NT_EQUAL2
          SUBB A, #30H
          LJMP  DISP
NT_EQUAL2:CJNE A,#33H,NT_EQUAL3
          SUBB A, #30H
          LJMP  DISP
NT_EQUAL3:CJNE A, #34H, NT_EQUAL4
          SUBB A, #30H
          LJMP  DISP
NT_EQUAL4:CJNE A, #35H, NT_EQUAL5
          SUBB A, #30H
          LJMP  DISP
NT_EQUAL5:CJNE A, #36H, NT_EQUAL6
          SUBB A, #30H
```

```
                LJMP  DISP
NT_EQUAL6:CJNE  A, #37H, NT_EQUAL7
                SUBB  A, #30H
                LJMP  DISP
NT_EQUAL7:CJNE  A, #38H, NT_EQUAL8
                SUBB  A, #30H
                LJMP  DISP
NT_EQUAL8:CJNE  A, #39H, DISP
                SUBB  A, #30H
DISP:           MOV   P1, A             ; send received data on P1
                SJMP  HERE
TRANS:          MOV   SBUF, A           ; transmit data to PC
HERE1:          JNB   TI, HERE1         ; wait for last bit to go
                CLR   TI                ; get ready for next character
                RET
                ORG   200H
DISPSTRING:     DB    '/8051 SERIAL COMMUNICATION/'
                DB    '     DEVELOPED BY: '
                DB    ' *ANAND ATALBIHARI BHASKAR* '
                DB    ' *Dr. H. N. Pandya*',0AH,0DH,0
                END
```

- **Front-End software for the PC**

The user-interface (Front-End) is used to select the available COM ports on the PC, receive the initializing data and to transmit data. The available ports can be determined using the GetPortNames() method of class SerialPort in System.IO.Ports namespace. The selected port is opened using Open() method of the instance of SerialPort. The baud rate, stop-bit, parity bit is defined while creating the instance itself. The initialization data string is displayed in the textbox automatically as the DataReceived() event is raised whenever any data is received in the buffer. The keystrokes transmitted to AT89C51 is sent using Write() method of SerialPort instance. The coding of the Front-End is as follows:

```
namespace serial{
   partial class frmSerial//Designer Form
   {
      /// <summary>
      /// Required designer variable.
      /// </summary>
      private System.ComponentModel.IContainer components = null;

      /// <summary>
      /// Clean up any resources being used.
      /// </summary>
      /// <param name="disposing">true if managed resources should be
disposed; otherwise, false.</param>
      protected override void Dispose(bool disposing)
```

```
        {
            if (disposing && (components != null))
            {
                components.Dispose();
            }
            base.Dispose(disposing);
        }

        #region Windows Form Designer generated code

        /// <summary>
        /// Required method for Designer support - do not modify
        /// the contents of this method with the code editor.
        /// </summary>
        private void InitializeComponent()
        {
            System.ComponentModel.ComponentResourceManager resources =
new
System.ComponentModel.ComponentResourceManager(typeof(frmSerial));
            this.groupBox1 = new System.Windows.Forms.GroupBox();
            this.Open = new System.Windows.Forms.Button();
            this.PortcomboBox = new System.Windows.Forms.ComboBox();
            this.groupBox2 = new System.Windows.Forms.GroupBox();
            this.Label3 = new System.Windows.Forms.Label();
            this.PictureBox2 = new System.Windows.Forms.PictureBox();
            this.txdtextBox = new System.Windows.Forms.TextBox();
            this.label2 = new System.Windows.Forms.Label();
            this.rxdtextBox = new System.Windows.Forms.TextBox();
            this.label1 = new System.Windows.Forms.Label();
            this.pictureBox1 = new System.Windows.Forms.PictureBox();
            this.groupBox1.SuspendLayout();
            this.groupBox2.SuspendLayout();

((System.ComponentModel.ISupportInitialize)(this.PictureBox2)).BeginInit();

((System.ComponentModel.ISupportInitialize)(this.pictureBox1)).BeginInit();
            this.SuspendLayout();
            //
            // groupBox1
            //
            this.groupBox1.Controls.Add(this.Open);
            this.groupBox1.Controls.Add(this.PortcomboBox);
            this.groupBox1.Font = new System.Drawing.Font("Verdana", 9.75F,
System.Drawing.FontStyle.Bold, System.Drawing.GraphicsUnit.Point,
((byte)(0)));
            this.groupBox1.Location = new System.Drawing.Point(131, 21);
            this.groupBox1.Name = "groupBox1";
            this.groupBox1.Size = new System.Drawing.Size(259, 100);
            this.groupBox1.TabIndex = 1;
            this.groupBox1.TabStop = false;
```

```
            this.groupBox1.Text = "Select COM Port";
            //
            // Open
            //
            this.Open.FlatStyle = System.Windows.Forms.FlatStyle.Flat;
            this.Open.Font = new System.Drawing.Font("Verdana", 12F,
System.Drawing.FontStyle.Regular, System.Drawing.GraphicsUnit.Point,
((byte)(0)));
            this.Open.Location = new System.Drawing.Point(153, 36);
            this.Open.Name = "Open";
            this.Open.Size = new System.Drawing.Size(75, 36);
            this.Open.TabIndex = 1;
            this.Open.Text = "Open";
            this.Open.UseVisualStyleBackColor = true;
            this.Open.Click += new System.EventHandler(this.Openp_Click);
            //
            // PortcomboBox
            //
            this.PortcomboBox.DropDownStyle =
System.Windows.Forms.ComboBoxStyle.DropDownList;
            this.PortcomboBox.FlatStyle =
System.Windows.Forms.FlatStyle.Popup;
            this.PortcomboBox.FormattingEnabled = true;
            this.PortcomboBox.Location = new System.Drawing.Point(12, 42);
            this.PortcomboBox.MaxDropDownItems = 4;
            this.PortcomboBox.Name = "PortcomboBox";
            this.PortcomboBox.Size = new System.Drawing.Size(109, 24);
            this.PortcomboBox.Sorted = true;
            this.PortcomboBox.TabIndex = 0;
            this.PortcomboBox.TabStop = false;
            this.PortcomboBox.SelectionChangeCommitted += new
System.EventHandler(this.PortcomboBox_SelectionChangeCommitted);
            //
            // groupBox2
            //
            this.groupBox2.Controls.Add(this.Label3);
            this.groupBox2.Controls.Add(this.PictureBox2);
            this.groupBox2.Controls.Add(this.txdtextBox);
            this.groupBox2.Controls.Add(this.label2);
            this.groupBox2.Controls.Add(this.rxdtextBox);
            this.groupBox2.Controls.Add(this.label1);
            this.groupBox2.Font = new System.Drawing.Font("Verdana", 9.75F,
System.Drawing.FontStyle.Bold, System.Drawing.GraphicsUnit.Point,
((byte)(0)));
            this.groupBox2.Location = new System.Drawing.Point(23, 136);
            this.groupBox2.Name = "groupBox2";
            this.groupBox2.Size = new System.Drawing.Size(367, 230);
            this.groupBox2.TabIndex = 2;
            this.groupBox2.TabStop = false;
            this.groupBox2.Text = "Serial Interface Data";
```

```
            //
            // Label3
            //
            this.Label3.AutoSize = true;
            this.Label3.Font = new System.Drawing.Font("Verdana", 9.75F,
System.Drawing.FontStyle.Italic, System.Drawing.GraphicsUnit.Point,
((byte)(0)));
            this.Label3.ImageAlign = System.Drawing.ContentAlignment.TopLeft;
            this.Label3.Location = new System.Drawing.Point(197, 208);
            this.Label3.Name = "Label3";
            this.Label3.Size = new System.Drawing.Size(164, 16);
            this.Label3.TabIndex = 9;
            this.Label3.Text = "Press keyboard to send";
            this.Label3.TextAlign =
System.Drawing.ContentAlignment.MiddleRight;
            //
            // PictureBox2
            //
            this.PictureBox2.Image =
global::serial.Properties.Resources.J0293236;
            this.PictureBox2.Location = new System.Drawing.Point(155, 195);
            this.PictureBox2.Name = "PictureBox2";
            this.PictureBox2.Size = new System.Drawing.Size(26, 29);
            this.PictureBox2.SizeMode =
System.Windows.Forms.PictureBoxSizeMode.Zoom;
            this.PictureBox2.TabIndex = 8;
            this.PictureBox2.TabStop = false;
            //
            // txdtextBox
            //
            this.txdtextBox.BorderStyle =
System.Windows.Forms.BorderStyle.FixedSingle;
            this.txdtextBox.Font = new System.Drawing.Font("Verdana", 12F,
System.Drawing.FontStyle.Regular, System.Drawing.GraphicsUnit.Point,
((byte)(0)));
            this.txdtextBox.Location = new System.Drawing.Point(19, 159);
            this.txdtextBox.Name = "txdtextBox";
            this.txdtextBox.Size = new System.Drawing.Size(332, 27);
            this.txdtextBox.TabIndex = 3;
            this.txdtextBox.KeyDown += new
System.Windows.Forms.KeyEventHandler(this.txdtextBox_KeyDown);
            //
            // label2
            //
            this.label2.AutoSize = true;
            this.label2.Font = new System.Drawing.Font("Verdana", 12F,
System.Drawing.FontStyle.Regular, System.Drawing.GraphicsUnit.Point,
((byte)(0)));
            this.label2.Location = new System.Drawing.Point(91, 138);
            this.label2.Name = "label2";
```

```
            this.label2.Size = new System.Drawing.Size(197, 18);
            this.label2.TabIndex = 2;
            this.label2.Text = "Enter Data to Transmit";
            //
            // rxdtextBox
            //
            this.rxdtextBox.BackColor = System.Drawing.Color.MistyRose;
            this.rxdtextBox.BorderStyle =
System.Windows.Forms.BorderStyle.FixedSingle;
            this.rxdtextBox.Font = new System.Drawing.Font("Courier New", 12F,
System.Drawing.FontStyle.Regular, System.Drawing.GraphicsUnit.Point,
((byte)(0)));
            this.rxdtextBox.Location = new System.Drawing.Point(19, 45);
            this.rxdtextBox.Multiline = true;
            this.rxdtextBox.Name = "rxdtextBox";
            this.rxdtextBox.ReadOnly = true;
            this.rxdtextBox.ScrollBars = System.Windows.Forms.ScrollBars.Both;
            this.rxdtextBox.Size = new System.Drawing.Size(332, 79);
            this.rxdtextBox.TabIndex = 1;
            this.rxdtextBox.TabStop = false;
            //
            // label1
            //
            this.label1.AutoSize = true;
            this.label1.FlatStyle = System.Windows.Forms.FlatStyle.Flat;
            this.label1.Font = new System.Drawing.Font("Verdana", 12F,
System.Drawing.FontStyle.Regular, System.Drawing.GraphicsUnit.Point,
((byte)(0)));
            this.label1.Location = new System.Drawing.Point(52, 23);
            this.label1.Name = "label1";
            this.label1.Size = new System.Drawing.Size(294, 18);
            this.label1.TabIndex = 0;
            this.label1.Text = "Data Received from Microcontroller";
            //
            // pictureBox1
            //
            this.pictureBox1.Image = global::serial.Properties.Resources.logo;
            this.pictureBox1.Location = new System.Drawing.Point(23, 23);
            this.pictureBox1.Name = "pictureBox1";
            this.pictureBox1.Size = new System.Drawing.Size(90, 98);
            this.pictureBox1.SizeMode =
System.Windows.Forms.PictureBoxSizeMode.Zoom;
            this.pictureBox1.TabIndex = 0;
            this.pictureBox1.TabStop = false;
            //
            // frmSerial
            //
            this.AutoScaleMode = System.Windows.Forms.AutoScaleMode.None;
            this.AutoSizeMode =
System.Windows.Forms.AutoSizeMode.GrowAndShrink;
```

```
            this.ClientSize = new System.Drawing.Size(414, 388);
            this.Controls.Add(this.groupBox2);
            this.Controls.Add(this.groupBox1);
            this.Controls.Add(this.pictureBox1);
            this.Font = new System.Drawing.Font("Verdana", 9F,
System.Drawing.FontStyle.Regular, System.Drawing.GraphicsUnit.Point,
((byte)(0)));
            this.FormBorderStyle =
System.Windows.Forms.FormBorderStyle.FixedSingle;
            this.Icon =
((System.Drawing.Icon)(resources.GetObject("$this.Icon")));
            this.Margin = new System.Windows.Forms.Padding(4);
            this.MaximizeBox = false;
            this.MaximumSize = new System.Drawing.Size(420, 420);
            this.MinimumSize = new System.Drawing.Size(420, 420);
            this.Name = "frmSerial";
            this.SizeGripStyle = System.Windows.Forms.SizeGripStyle.Show;
            this.StartPosition =
System.Windows.Forms.FormStartPosition.CenterScreen;
            this.Text = "Serial Communicator for Microcontrollers";
            this.TopMost = true;
            this.TransparencyKey =
System.Drawing.Color.FromArgb(((int)(((byte)(192)))), ((int)(((byte)(255)))),
((int)(((byte)(255)))));
            this.Load += new System.EventHandler(this.frmSerial_Load);
            this.groupBox1.ResumeLayout(false);
            this.groupBox2.ResumeLayout(false);
            this.groupBox2.PerformLayout();

((System.ComponentModel.ISupportInitialize)(this.PictureBox2)).EndInit();

((System.ComponentModel.ISupportInitialize)(this.pictureBox1)).EndInit();
            this.ResumeLayout(false);

        }


        #endregion

        private System.Windows.Forms.PictureBox pictureBox1;
        private System.Windows.Forms.GroupBox groupBox1;
        private System.Windows.Forms.ComboBox PortcomboBox;
        private System.Windows.Forms.GroupBox groupBox2;
        private System.Windows.Forms.TextBox rxdtextBox;
        private System.Windows.Forms.Label label1;
        internal System.Windows.Forms.PictureBox PictureBox2;
        private System.Windows.Forms.TextBox txdtextBox;
        private System.Windows.Forms.Label label2;
        private System.Windows.Forms.Button Open;
        internal System.Windows.Forms.Label Label3;
```

```
    }
}

//In frmSerial.cs
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Text;
using System.Windows.Forms;
using System.IO.Ports;
namespace serial
{
    public partial class frmSerial : Form
    {
        SerialPort com1;
        //gets list of ports
        protected string[] ports = SerialPort.GetPortNames();
        public static string p;
        protected string rxd;
        public frmSerial()
        {
            InitializeComponent();
        }
         private void frmSerial_Load(object sender, EventArgs e)
        {
            MessageBox.Show("First select & open the port, then start kit");
            foreach (string port in ports)
            {
                this.PortcomboBox.Items.Add(port);
            }
        }
        //Sets select port
        void PortcomboBox_SelectionChangeCommitted(object sender,
System.EventArgs e)
        {
            p = this.PortcomboBox.SelectedItem.ToString();
            com1 = new SerialPort(p, 9600, Parity.None, 8, StopBits.One);
            this.com1.DataReceived += new
SerialDataReceivedEventHandler(com1_DataReceived);
        }
        //Opens the selected port on click of open button
        private void Openp_Click(object sender, EventArgs e)
        {
            com1.Open();
            if (com1.IsOpen == true)
            {
                this.Open.Enabled = false;
            }
```

```
    }
      //This event is ready when data is received
      public void com1_DataReceived(object sender,
SerialDataReceivedEventArgs e)
      {
         try
         {
            this.rxdtextBox.Text = com1.ReadLine();
         }
         catch (InvalidOperationException i){
            MessageBox.Show(i.ToString());
         }
         catch (System.IO.IOException io) {
            MessageBox.Show(io.ToString());
         }
      }
      //Sends the Keystroke to microcontroler
      private void txdtextBox_KeyDown(object sender, KeyEventArgs e)
      {//processes 0 to 9 characters only
         char[] ch = new char[2];
         ch.SetValue((char)e.KeyCode, 1);
         com1.Write(ch, 1, 1);
         }
      }
}
```

//The main Program.cs

```
using System;
using System.Collections.Generic;
using System.Windows.Forms;
using System.Threading;
namespace serial
{
   static class Program
   {
      /// <summary>
      /// The main entry point for the application.
      /// </summary>
      [STAThread]
      static void Main()
      {
         Application.EnableVisualStyles();
         Application.SetCompatibleTextRenderingDefault(false);
         Application.Run(new frmSerial());
      }
   }
}
```

Installation of the Software

The software requires .NET Framework 2.0. The necessary Framework is supplied with the software or may be available from www.microsoft.com freely. The software is installed using ClickOnce application deployment method. This makes it easy to install the software. The choice of software development using .NET technology is to provide a code-execution environment that promotes safe execution of the code, including code created by an unknown or semi-trusted third party. Programmers can write application in their development language of choice; take the full advantage of the runtime, the class library, and the components written in other languages by other developers.
We are providing the setup.exe for the full installation of the front-end software. Figure 10-6 shows the initialization message of the front-end and figure 10-7 shows the startup screen for the front-end.

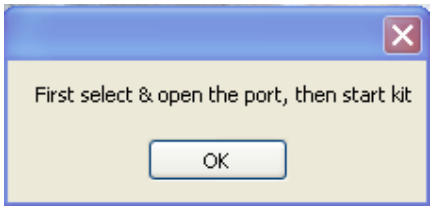


Figure 10-6: Initialization message for the front-end.

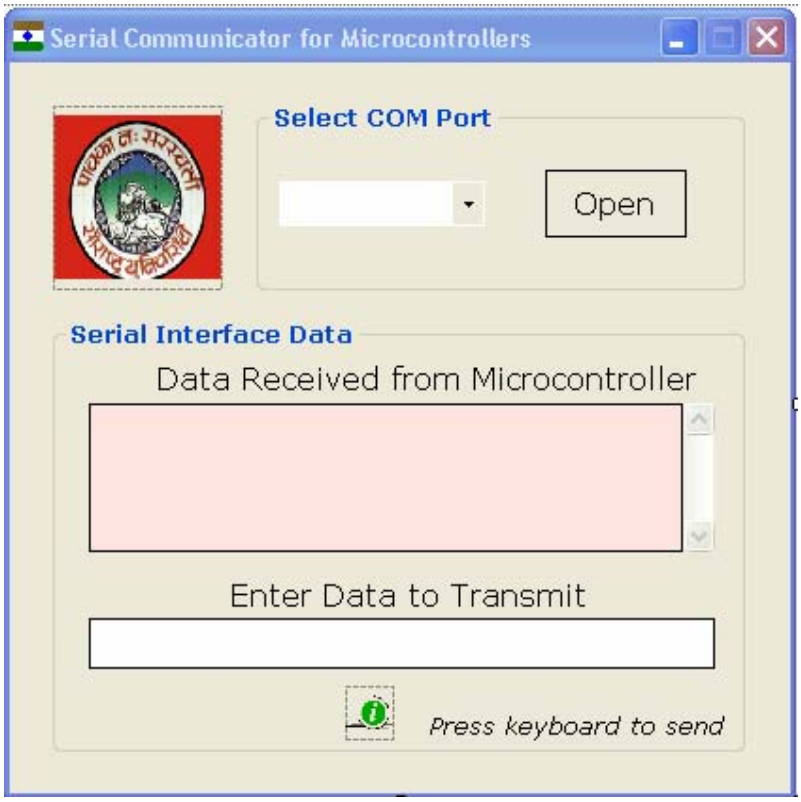


Figure 10-7: Screenshot of the startup screen for the front end.

Press OK to the first message. Then click on “Select Com Port” item of Figure-7. Pull down menu will appear, select proper COM port (The port which you connected the hardware). Then power on the hardware. This will display initialization message “/8051 SERIAL COMMUNICATION/ DEVELOPED BY: *ANAND ATALBIHARI BHASKAR* *Dr. H. N. Pandya*” in the box titled “Data Received from Microcontroller”. The cursor will be set by default in the box “Enter Data to Transmit”. Now press any key from the keyboard, the character of the key pressed will be display in this box and at the same time its equivalent ASCII value in binary will be display on the LED’s of the hardware.

Component layout diagram

Figure 10-8 shows the component side of PCB for the Microcontroller Serial Communicator Simplified.

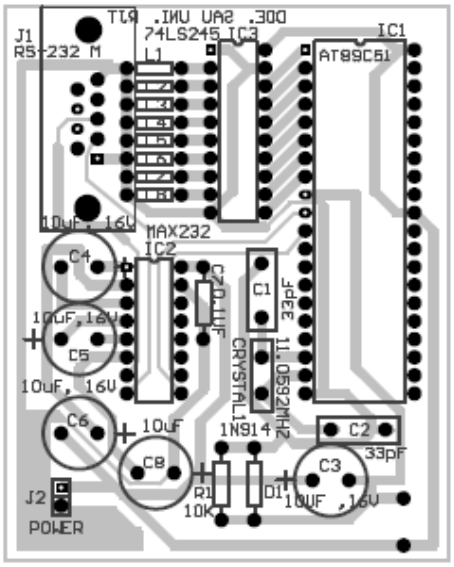


Figure 10-8: Actual size component side PCB for the Microcontroller Serial Communicator Simplified.

Solder Side PCB for Serial Communicator

Figure 10-9 shows the actual size single side PCB for the Microcontroller Serial Communicator Simplified.

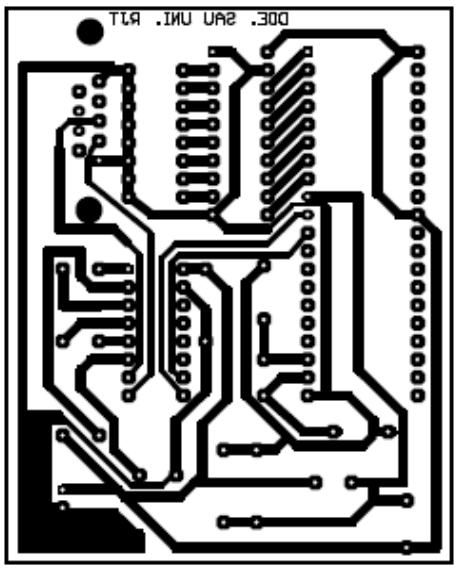


Figure 10-9: Actual size single side PCB for the Microcontroller Serial Communicator Simplified.

10.5 Future Developments

Microcontroller serial communicator simplifies the RS232 serial communication interface (SCI) between the programmable system and PC. Only one UART is available on the SU51MB as presented in previous chapter this can be improved to two. Serial terminals can made to communicate without any external circuitry using serial link only by using microcontroller on its base. More work can be done on full duplex mode of communication.

Section - IV: Design of the microcontroller based embedded circuits and system.

Section 4: Design of the microcontroller based Embedded Circuits and systems.

Chapter 11 MCS51 Based Embedded System: RF Test Kit and Its Applications

11.1 Introduction

The modern age is making more and more advancement in the field of the wireless automation control. Present work is to provide small, simple and cost-effective wireless equipment control for home appliances or industrial instrumentation control with the help of microcontroller. This will definitely be a stepping stone for the beginner or hobbyist to start interfacing microcontrollers with RF-modules to design and develop applications based on wireless link. A simple RF Test kit is developed and implemented in the Wireless Equipment Control application.

In this application, at a time four devices timeout can be controlled using wireless link. The LCD module is initialized first to give user interface to set time-out of the remote devices connected with the receiver module. As soon as the time-out is set, all the four devices will be turned on initially and according to the time-out set for a particular device it will be turned off. For current application the time-out is limited with the range 00 to 99 seconds which can be easily improved by doing modification in the delay subroutine of the software provided. Hence, a simple wireless automation control system is discussed with the help of small RF Test Kit, which can easily be fixed at home as well as in industry for the radio control of devices. The block diagram of the system is shown in the Figure 11-1.

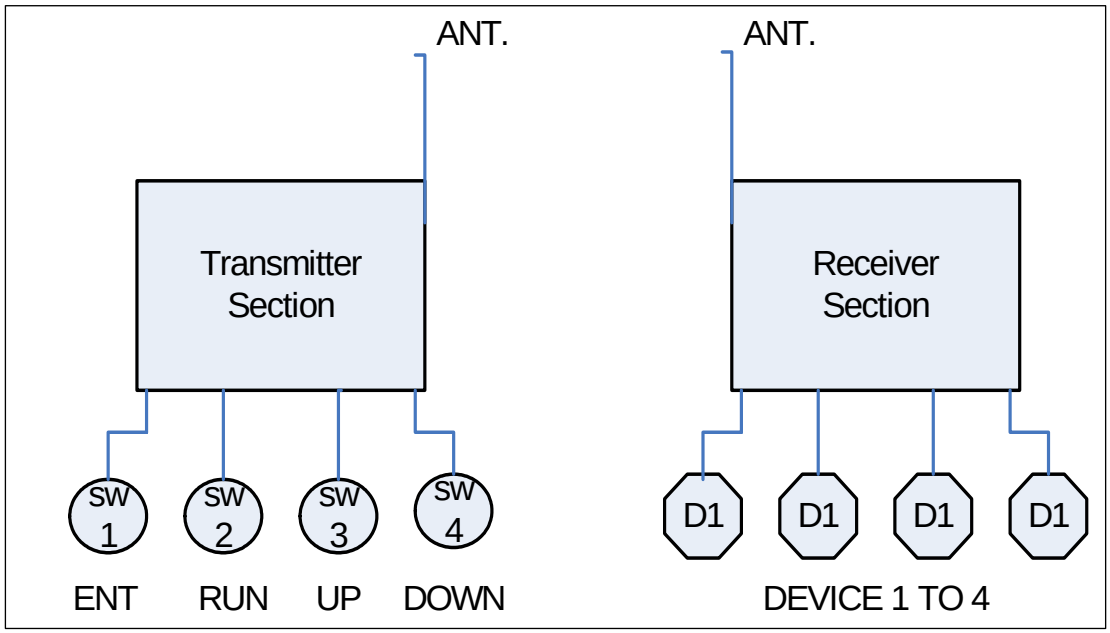


Figure 11-1: Basic block diagram of the system.

11.2 RF Test Kit and Wireless Equipment Control Basics:

For the remote control of any gadget, one needs an RF transmitter and a receiver. The main specification of the wireless link (in terms of RF modules used) is given in the Table–11.1.

Main Specifications	
1. Frequency of operation	434 MHz
2. Modulation	ASK
3. Range	30 Feet
4. Power Supply	12 volts (Rx) 9 volts to 12 volts (Tx)

Table–11.1: RF module specifications.

1. RF Test Kit Transmitter Section

The Figure 11-2 shows the block diagram of the transmitter section.

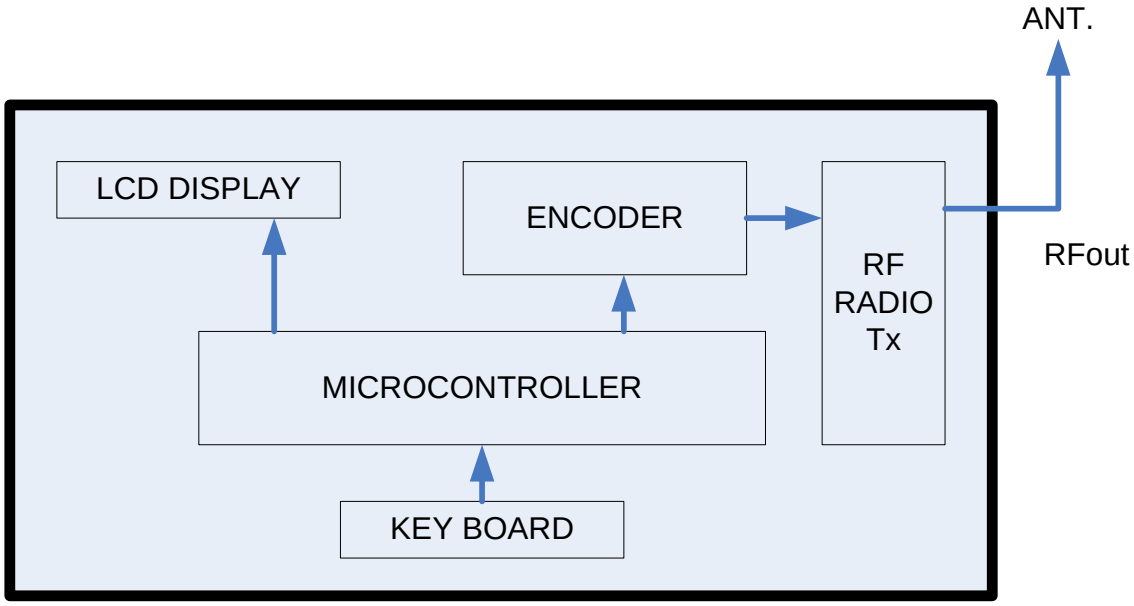


Figure 11-2: Block diagram of RF test kit transmitter section.

Transmitter section consists of RF module, Encoder IC, LCD display, Microcontroller and Keyboard.

- Key Board:

Four push button switches are used in the transmitter section. When these push button switches are open, logic “0” is constantly fed to the port pins of the microcontroller. The naming convention for this work to these push button switches is UP, DOWN, ENTER and RUN. When one of the buttons is pressed, logic one is sent to the microcontroller. UP and DOWN buttons are used to set time (in seconds) for ON time of the device connected with the receiver section. ENER key is used to switch between the devices time set in the LCD module. Once the user entered the time for the device he or she must press ENTER key to go to next time set for the device. So the ENTER key is pressed to go in the next function of programming flow. When the time-

out for all the four devices is set, RUN key is used. RUN key is pressed to run the device control program, which transfers the timeout data over the wireless link to the receiver section. The device control program is stored in the microcontroller's memory and it will OFF the device as per the time-out set by the user.

- Microcontroller:

The microcontroller is the central controlling part of transmitter section which controls the all related electronics. For the current work the SU51MB microcontroller board is utilized as base of the application. The board is connected with LCD module, keyboard, and encoder IC (HT12E). All the detailing of the SU51MB is provided in the Chapter-4. Microcontroller reads the input data by means of the push to on switches, which is connected to port 2 (pin21 to pin24) and at the same time it displays on the LCD module interfaced with microcontroller. In microcontroller software is programmed to control ports input and output data. Port3 provides the read data to the encoder IC HT12E (pin10 to 14).

- LCD Module:

LCD module used for this embedded application is 2-line 16-character LCD module. LCD displays the status of the main program which is running inside the microcontroller.

- Encoder:

The HT12E is an encoder IC. The D0-D3 data inputs (pins10 to pin13) to the encoder have been connected to the microcontroller as shown in the transmitter block diagram. The eight inputs to the IC (pin1 to pin8) are called the address inputs. These tell the encoder which address to send. The decoder IC HT12D also has these inputs – the address on both ICs must match for the data to be valid.
Each address pin may be connected to ground or 5V. In this project they are connected to 5V, so the address is set to 255d (in decimal) on both ICs. If one is going to connect all the address pins to ground, then the address would be 000d. Thus there are 256 possible addresses available. So, one can set up switches to control one or more of the encoder address pins, and then have several decoders, each set to a different address.
There are two more pins on the encoder. Pin14 is an input pin called Transmit Enable (TE). Only when it is connected to ground, the encoder will send a transmission. Whenever a button is pressed, logic '0' is sent to this pin through microcontroller thus making it active and enabling transmission.
Pin17 is data out (D_{out}) which sends the serial stream of pulses containing the address and data. This is connected to the data input of the radio transmitter module. The detailed working of the encoder IC can be understood easily by means of the flowchart shown in the Figure 11-3 below.

- Address Select:

There is a facility made available to the user to select his own unique address from a range of 000d to 255d. For this purpose, the address lines are pulled out and kept between the two columns of Vcc and Gnd. Shorting the corresponding address line to either Vcc or Gnd does the selection.

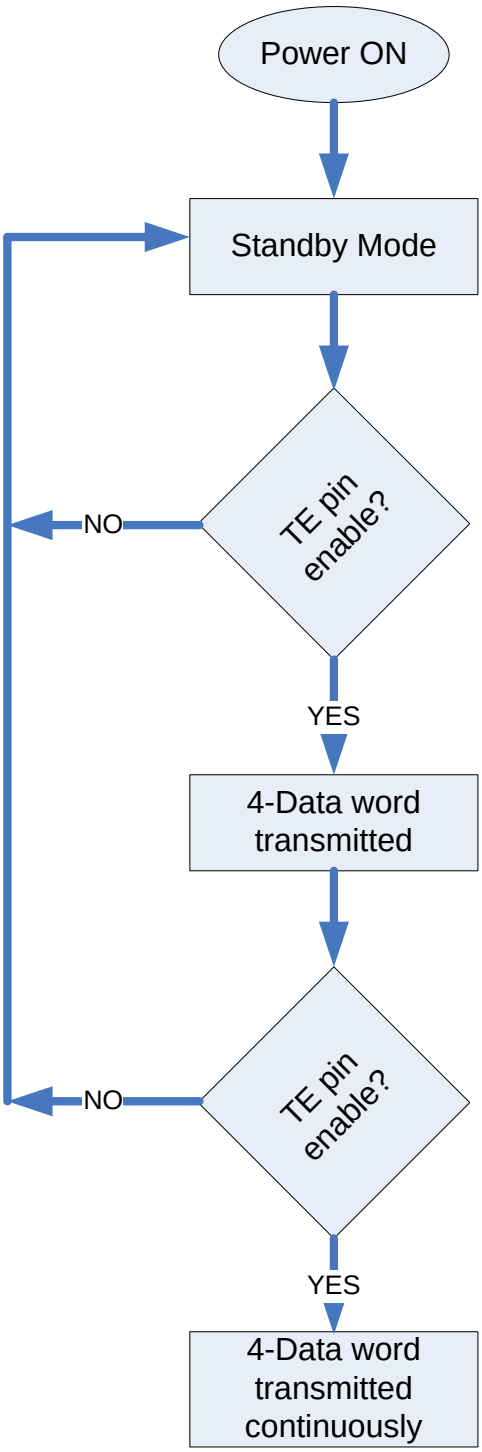


Figure11-3: Working of an encoder IC.

- Radio Transmitter Module:

The radio transmitter module uses digital modulation technique called Amplitude Shift Keying (ASK) or On-Off keying. In this technique, whenever there is LOGIC '1' to be sent; only for that time the carrier is sent else nothing is sent for LOGIC '0'. This modulated signal is then transmitted using the antenna. The wave forms in Figure 11-4 clears the ASK concept.

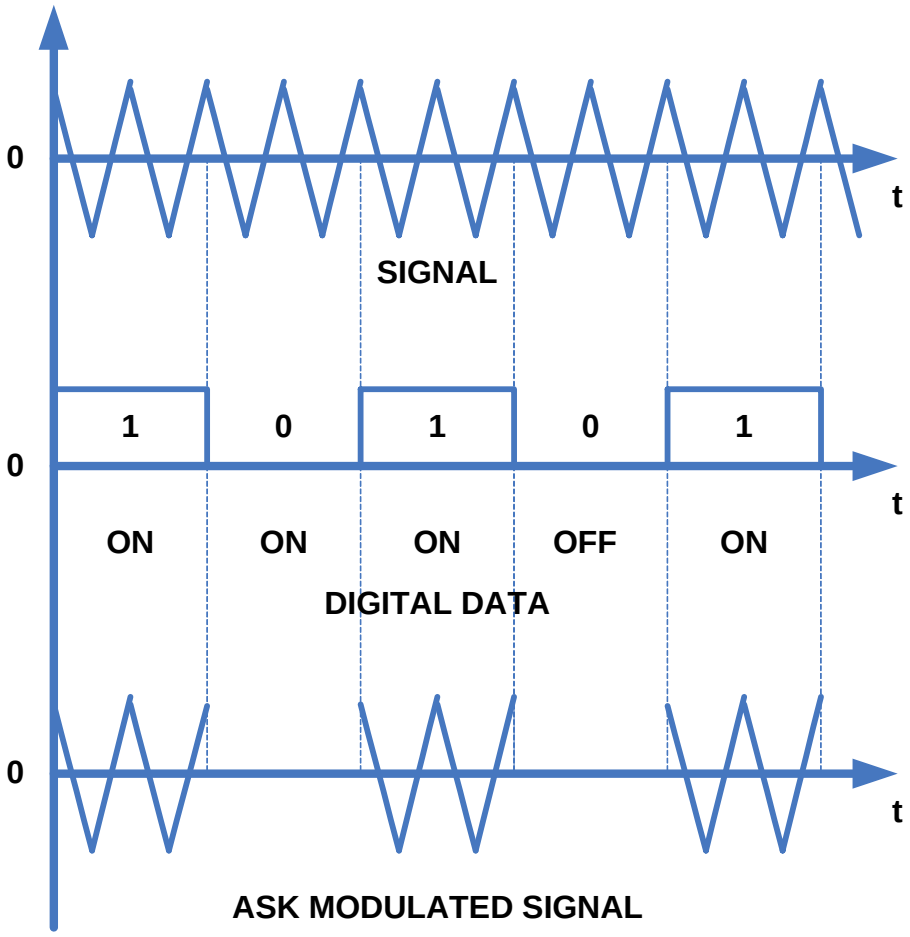


Figure 11-4: ASK for radio transmitter module.

2. RF Test Kit Receiver Section

This unit is having a range as long as 30 feet and working efficiently within this range without any problem. If a proper antenna is used than the range may be increased up to 30 meters. Figure 11-5 shows the block diagram of the receiver section.

As shown in the block diagram the receiver section comprises of a radio receiver module, decoder IC, multiplexer and mode select block, and relay driver circuitry.

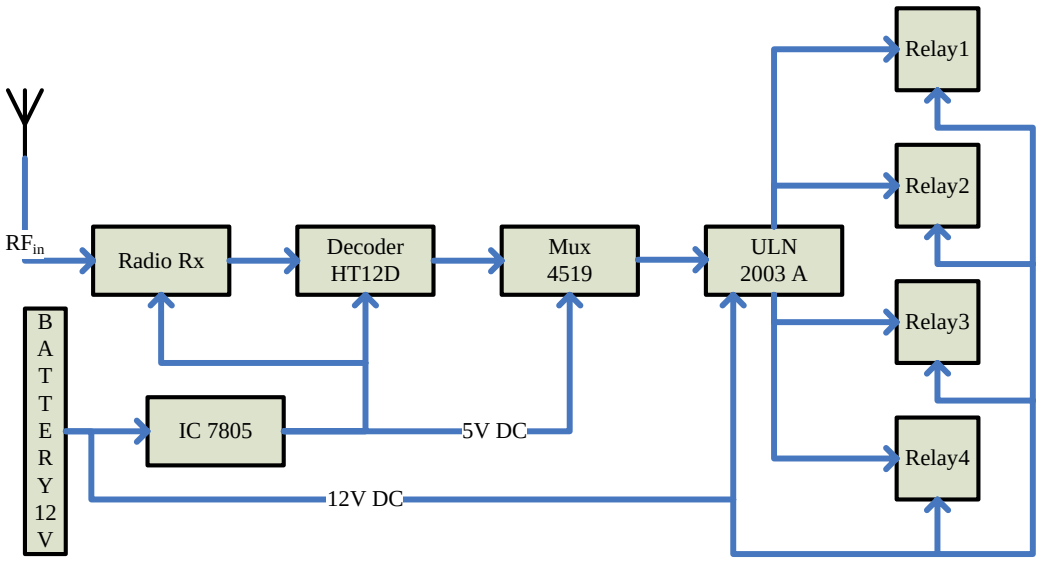


Figure 11-5: Block diagram of the RF Test Kit receiver section.

- RF Module:

Radio receiver module receives the ASK signal and demodulates it to get back the address bits and the data bits.

- Decoder HT12D IC:

HT12D is a decoder IC. The D_{in} pin receives the address bits and data bits serially from the RF module. The D0-D3 data pins (pins10 to 13) of the decoder then send the suitable four bit data.
There are eight inputs to the IC (pin1 to pin8) called the address inputs. These tell the decoder which address to acknowledge upon receiving one from a transmitter. The address on both the encoder and decoder ICs must match for the data to be valid. Each address pin may be connected to ground or 5v. In this project they have been connected to 5v, so the address is set to 255d on both ICs. If you were to connect all the address pins to ground, the address would be 000d. Thus there are 256 possible addresses available. Thus user can set up switches to control one or more of the encoder address pins, and then have several decoders, each set to a different address. The VT pin is connected to the Latch/Momentary mode select section and is also made available on the relay side so as to switch on the relay whenever the VT pin is active high.

- Address Select:

There is a facility made available to the user to select his own unique address from a range of 000d to 255d. For this purpose, the address lines are pulled out and kept between the two columns of Vcc and Ground. In the presented work all address lines are kept on the Vcc columns. Both the address in transmitter and receiver will be the same for RF communication.

- Latch/Momentary Mode Select:

This setting allows the selection of either latch/momentary mode. In latch mode, the data outputs from the decoder are latched and the moment the VT pin is active low, the action stops. In momentary mode, the data outputs are not latched. Shorting appropriate pins using a jumper makes selection. In momentary mode, as long as the VT pin is active high, the action continues. The same particular action keeps repeating.

- Invert/Normal Mode Select:

This section allows user to select between normal output and inverted output. Shorting appropriate pins using a jumper makes selection.

- Multiplexer IC 4519:

Integrated circuit 4519 is a QUAD common addressable 2 to 1 multiplexer. The select input comes from the latch/momentary mode select section. There are eight input lines of which four are permanently connected to the invert/normal mode selection section, while the rest four are connected to the data output lines of the decoder IC. When the latch mode is selected, whatever data present at the output pins is latched i.e. they remain the same and the respective relay is ON forever till the next change in the mode selection. Whenever momentary mode is selected, whatever data present at the output pins is available as long as the VT pin remains active high. As soon as the VT pin becomes active low, the respective relay is switched OFF. In the invert mode, the output of the multiplexer is active low, while in normal mode it is active high.

- Relay Driver:

ULN 2003A is seven array driver IC. This chip has an in-built relay driver circuitry and hence no additional circuitry is required.

11.3 Circuit Diagram and its Explanation

The circuit diagram of the wireless equipment control using microcontroller is divided into two sections as per the system block diagram into transmitter section and receiver section. Figure 11-6 shows schematic of the transmitter section of the system. Transmitter section circuit consists of microcontroller (AT89C51), RF module interface (433 MHz ASK transmitter), LCD module interface (2X16 character) circuitry and four keys (push button) S1, S2, S3, and S4. The keys are providing the functions UP, DOWN, ENT and RUN. By using these key one can easily set the time-out for the devices to turn them ON/OFF.

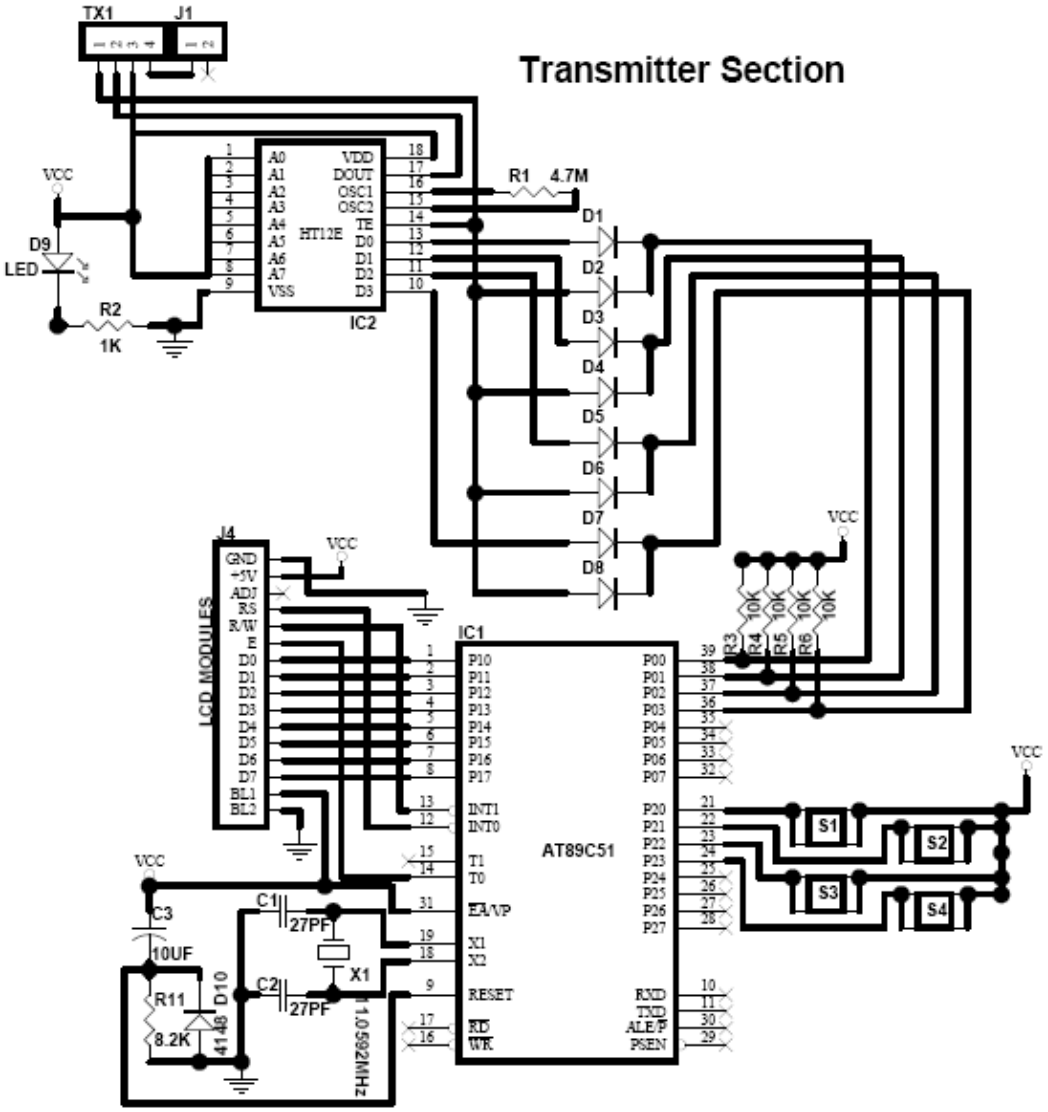


Figure 11-6: Circuit Diagram for the transmitter section.

The default time for all devices is 00 seconds. So using UP key one can increment a time range of one second and by DOWN key one can decrement time one second down. At the same time LCD module shows the current status of the increments and decrements. When the time out for the one device is set user is supposed to press the ENT key so the control transfers to set the time-out for the next device. In the same way all the four devices time-out must be set before pressing the final RUN key. When the RUN key is pressed it will execute the device control program subroutine which will automatically collect all information of time-out entered by the user and sends the processed data to the encoder IC HT12E. Encoder IC mixes the data with the address and sends to the data pin of the RF transmitter module, which transmit mixed information through antenna. To transmit data or information RF module uses Amplitude shift keying (ASK) modulation technique.

Figure 11-7 shows the receiver circuit of the system. Receiver circuit consist RF module (433 MHz ASK receiver), decoder IC (HT12D), multiplexer IC (MN4519B) and relay driver IC (ULN2003A).

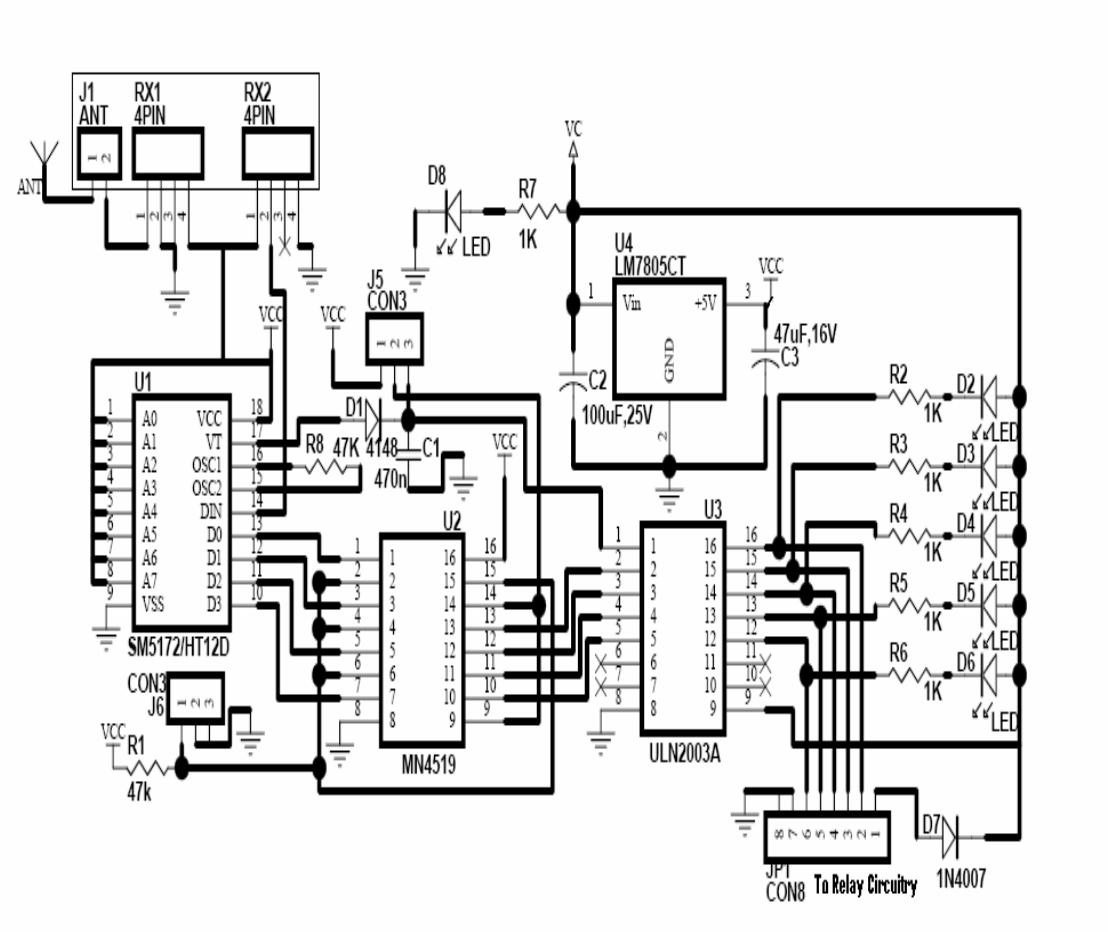


Figure 11-7: Circuit diagram for receiver section.

The RF receiver module will receive signal which is transmitted by the transmitter. IC HT12D will separate data and address from the received information. Data is accepted only when received address is matched with address assigned to the both encoder and decoder IC. In present case I have used 1111 as the permanent address for the communication. The separated data ID then fed to the multiplexer IC MN4519B which multiplexes and latches the input data. The latched output data is fed to the relay driver IC ULN2003, which is used to drive the relay. Connected device with relay becomes ON or OFF according to the received data.

11.4 Software Flowchart

The software programmed in the transmitter section microcontroller is shown below in Figure 11-8.

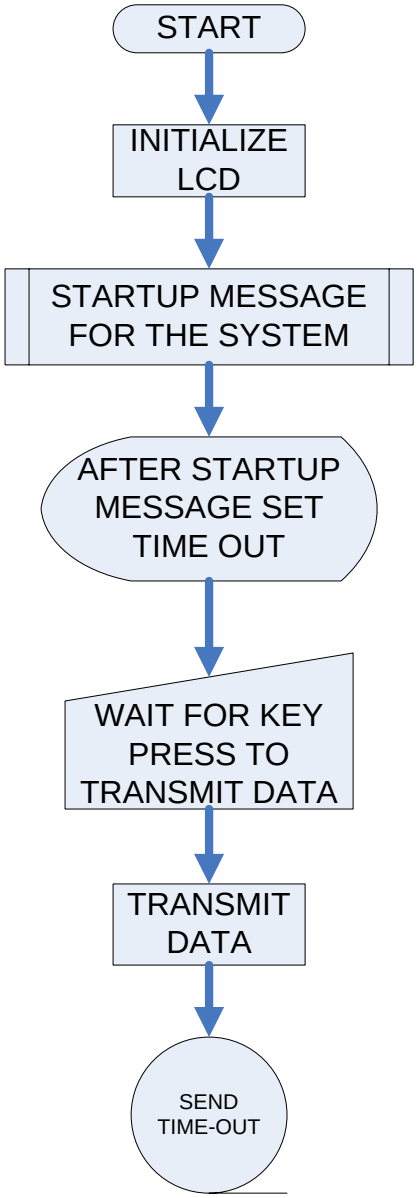


Figure 11-8: Software Flowchart for RF test kit.

11.5 Software Section

The software section is the heart of the system which is designed to provide the status as well as control to the devices. All the control and time-out for the devices is embedded within the AT89C51 microcontroller.
The software is responsible for identifying the key pressed and displaying the key code on the LCD module. Reading the codes and outputting the same to the proper port at proper time is done through proper instructions of the microcontroller. Here Port 0 is configured as output port which is interfaced with the RF module. Port 1 is used for LCD interface and Port 2 is used for the input where the push to on switches are connected. The program can be burnt in the AT89C51 microcontroller by using any standard programmer available in the market. In this case LabTool Universal programmer is used to program the microcontroller.

The software listing is given below:

```
;"Wireless Automation Control System" by AAB
; Main Program: Date: 10/04/2008
$MOD51
        DB0     EQU     P1.0
        DB1     EQU     P1.1
        DB2     EQU     P1.2
        DB3     EQU     P1.3
        DB4     EQU     P1.4
        DB5     EQU     P1.5
        DB6     EQU     P1.6
        DB7     EQU     P1.7
        EN      EQU     P3.4
        RW      EQU     P3.3
        RS      EQU     P3.2
        UP      EQU     P2.0
        DOWN    EQU     P2.1
        ENT     EQU     P2.2
        RUN     EQU     P2.3
        DATA1   EQU     P1
        PACK1   EQU     60H
        PACK2   EQU     61H
        PACK3   EQU     62H
        PACK4   EQU     63H
        RAM1    EQU     70H
        RAM2    EQU     71H
        RAM3    EQU     72H
        RAM4    EQU     73H
        RAM5    EQU     74H
        RAM6    EQU     75H
        RAM7    EQU     76H
        RAM8    EQU     77H
        ORG     00H
                LJMP  MAIN
```

```
        ORG     150H
MAIN:   SETB    P0.4
        MOV     P0,#1FH
        MOV     P0,#0FH
        SETB    P0.4
        MOV     P0,#10H
        MOV     P0,#00H
        SETB    P0.4
        LCALL   INIT_LCD        ;LCD Initialization
        MOV     A,#80H
        LCALL   CMD
        MOV     A,#'P'
        LCALL   WRITE_TEXT
        MOV     A,#'R'
        LCALL   WRITE_TEXT
        MOV     A,#'E'
        LCALL   WRITE_TEXT
        MOV     A,#'S'
        LCALL   WRITE_TEXT
        MOV     A,#'S'
        LCALL   WRITE_TEXT
        MOV     A,#' '
        LCALL   WRITE_TEXT
        MOV     A,#'A'
        LCALL   WRITE_TEXT
        MOV     A,#'N'
        LCALL   WRITE_TEXT
        MOV     A,#'Y'
        LCALL   WRITE_TEXT
        MOV     A,#' '
        LCALL   WRITE_TEXT
        MOV     A,#'K'
        LCALL   WRITE_TEXT
        MOV     A,#'E'
        LCALL   WRITE_TEXT
        MOV     A,#'Y'
        LCALL   WRITE_TEXT
        MOV     A,#'!'
        LCALL   WRITE_TEXT
        MOV     A,#'!'
        LCALL   WRITE_TEXT
        MOV     A,#'!'
        LCALL   WRITE_TEXT
        SETB    P0.4
        MOV     P2,#0FFH        ;P2 AS INPUT
KEEP:   MOV     A,P2            ;READ       P2
        ANL     A,#0FH
        JZ      KEEP
        LCALL   DELAY           ;Debounce check
        MOV     A,P2
```

```
ANL     A,#0FH
JZ      KEEP
LCALL   CLEAR_LCD
MOV     A,#80H      ;set cursor to first position.
LCALL   CMD
MOV     A,#'T'
LCALL   WRITE_TEXT
MOV     A,#'O'
LCALL   WRITE_TEXT
MOV     A,#' '
LCALL   WRITE_TEXT
MOV     A,#'S'
LCALL   WRITE_TEXT
MOV     A,#'E'
LCALL   WRITE_TEXT
MOV     A,#'T'
LCALL   WRITE_TEXT
MOV     A,#' '
LCALL   WRITE_TEXT
MOV     A,#'T'
LCALL   WRITE_TEXT
MOV     A,#'I'
LCALL   WRITE_TEXT
MOV     A,#'M'
LCALL   WRITE_TEXT
MOV     A,#'E'
LCALL   WRITE_TEXT
MOV     A,#' '
LCALL   WRITE_TEXT
MOV     A,#'O'
LCALL   WRITE_TEXT
MOV     A,#'U'
LCALL   WRITE_TEXT
MOV     A,#'T'
LCALL   WRITE_TEXT
NOP
NOP
NOP
NOP
NOP
NOP
NOP
MOV     A,#0C0H
LCALL   CMD
LCALL   WAIT_LCD
MOV     A,#'P'
LCALL   WRITE_TEXT
MOV     A,#'R'
LCALL   WRITE_TEXT
MOV     A,#'E'
```

```
                LCALL   WRITE_TEXT
                MOV     A,#'S'
                LCALL   WRITE_TEXT
                MOV     A,#'S'
                LCALL   WRITE_TEXT
                MOV     A,#' '
                LCALL   WRITE_TEXT
                MOV     A,#'!'
                LCALL   WRITE_TEXT
                MOV     A,#'E'
                LCALL   WRITE_TEXT
                MOV     A,#'N'
                LCALL   WRITE_TEXT
                MOV     A,#'T'
                LCALL   WRITE_TEXT
                MOV     A,#'!'
                LCALL   WRITE_TEXT
H1:             JB      ENT,G1
                SJMP    H1
G1:             LCALL   DELAY
                JB      ENT,T_SET_D
                SJMP    H1
T_SET_D:        LCALL   CLEAR_LCD
                MOV     A,#80H      ;set cursor to frist position.
                LCALL   CMD
                MOV     A,#'D'
                LCALL   WRITE_TEXT
                MOV     A,#'1'
                LCALL   WRITE_TEXT
                MOV     A,#'_'
                LCALL   WRITE_TEXT
                MOV     A,#'T'
                LCALL   WRITE_TEXT
                MOV     A,#'='
                LCALL   WRITE_TEXT
                MOV     A,#88H
                LCALL   CMD
                MOV     A,#'D'
                LCALL   WRITE_TEXT
                MOV     A,#'2'
                LCALL   WRITE_TEXT
                MOV     A,#'_'
                LCALL   WRITE_TEXT
                MOV     A,#'T'
                LCALL   WRITE_TEXT
                MOV     A,#'='
                LCALL   WRITE_TEXT
                MOV     A,#0C0H
                LCALL   CMD
                MOV     A,#'D'
```

```
                LCALL   WRITE_TEXT
                MOV     A,#'3'
                LCALL   WRITE_TEXT
                MOV     A,#'_'
                LCALL   WRITE_TEXT
                MOV     A,#'T'
                LCALL   WRITE_TEXT
                MOV     A,#'='
                LCALL   WRITE_TEXT
                MOV     A,#0C8H
                LCALL   CMD
                MOV     A,#'D'
                LCALL   WRITE_TEXT
                MOV     A,#'4'
                LCALL   WRITE_TEXT
                MOV     A,#'_'
                LCALL   WRITE_TEXT
                MOV     A,#'T'
                LCALL   WRITE_TEXT
                MOV     A,#'='
                LCALL   WRITE_TEXT
H2:             JB      ENT,G2
                SJMP    H2
G2:             LCALL   DELAY
                JB      ENT,STIME
                SJMP    H2
STIME:          MOV     A,#85H      ;SET CURSOR TO L=1,P=5
                LCALL   CMD
                LCALL   UP_DOWN
                MOV     RAM1,A              ;D_1
                MOV     A,#86H
                LCALL   CMD
                LCALL   WAIT_LCD
                LCALL   UP_DOWN
                MOV     RAM2,A              ;D_1
                MOV     A,#8DH
                LCALL   CMD
                LCALL   WAIT_LCD
                LCALL   UP_DOWN
                MOV     RAM3,A              ;D_2
                MOV     A,#8EH
                LCALL   CMD
                LCALL   WAIT_LCD
                LCALL   UP_DOWN
                MOV     RAM4,A              ;D_2
                MOV     A,#0C5H
                LCALL   CMD
                LCALL   WAIT_LCD
                LCALL   UP_DOWN
                MOV     RAM5,A              ;D_3
```

```
        MOV     A,#0C6H
        LCALL   CMD
        LCALL   WAIT_LCD
        LCALL   UP_DOWN
        MOV     RAM6,A          ;D_3
        MOV     A,#0CDH
        LCALL   CMD
        LCALL   WAIT_LCD
        LCALL   UP_DOWN
        MOV     RAM7,A          ;D_4
        MOV     A,#0CEH
        LCALL   CMD
        LCALL   WAIT_LCD
        LCALL   UP_DOWN
        MOV     RAM8,A
        MOV     P2,#0FFH
H3:     JB      RUN,G4
        SJMP    H3
G4:     LCALL   DELAY
        JB      RUN,RU1
        SJMP    H3
RU1:    MOV     A,RAM1
        SWAPA
        ANL     A,#0F0H
        MOV     B,A
        MOV     A,RAM2
        CLR     C
        SUBB    A,#30H
        ADD     A,B
        MOV     PACK1,A
        MOV     A,RAM3
        SWAP    A
        ANL     A,#0F0H
        MOV     B,A
        MOV     A,RAM4
        CLR     C
        SUBB    A,#30H
        ADD     A,B
        MOV     PACK2,A
        MOV     A,RAM5
        SWAP    A
        ANL     A,#0F0H
        MOV     B,A
        MOV     A,RAM6
        CLR     C
        SUBB    A,#30H
        ADD     A,B
        MOV     PACK3,A
        MOV     A,RAM7
        SWAP    A
```

```
        ANL     A,#0F0H
        MOV     B,A
        MOV     A,RAM8
        CLR     C
        SUBB    A,#30H
        ADD     A,B
        MOV     PACK4,A
        MOV     A,PACK1     ;DECI TO HEX SUB..FOR R0
        MOV     R4,A
        LCALL   D_T_H
        MOV     PACK1,A
        MOV     A,PACK2         ;FOR R1
        MOV     R4,A
        LCALL   D_T_H
        MOV     PACK2,A
        MOV     A,PACK3         ;FOR R2
        MOV     R4,A
        LCALL   D_T_H
        MOV     PACK3,A
        MOV     A,PACK4         ;FOR R3
        MOV     R4,A
        LCALL   D_T_H
        MOV     PACK4,A
        NOP
        MOV     R0,#99
        MOV     R6,#00H     ;DEVICE CONTROL SUB
        MOV     P0,#0FH     ;All Devices On     p1=00
L1:     LCALL   TIMER
        INC     R6
        MOV     A,R6
        CJNE    A,PACK1,D1
        CLR     P0.0
        MOV     A,#85H          ;SET CURSOR TO L
        LCALL   CMD
        MOV     A,#24H
        LCALL   WRITE_TEXT
        MOV     A,#24H
        LCALL   WRITE_TEXT
        MOV     A,R6
D1:     CJNE    A,PACK2,D2
        CLR     P0.1
        MOV     A,#8DH          ;SET CURSOR TO L
        LCALL   CMD
        MOV     A,#24H
        LCALL   WRITE_TEXT
        MOV     A,#24H
        LCALL   WRITE_TEXT
        MOV     A,R6
D2:     CJNE    A,PACK3,D3
        CLR     P0.2
```

```
                MOV     A,#0C5H         ;SET CURSOR TO L
                LCALL   CMD
                MOV     A,#24H
                LCALL   WRITE_TEXT
                MOV     A,#24H
                LCALL   WRITE_TEXT
                MOV     A,R6
D3:             CJNE    A,PACK4,D4
                CLR     P0.3
                MOV     A,#0CDH     ;SET CURSOR TO L=1,P=5
                LCALL   CMD
                MOV     A,#24H
                LCALL   WRITE_TEXT
                MOV     A,#24H
                LCALL   WRITE_TEXT
                MOV     A,R6
D4:             DJNZ    R0,L1
                MOV     P0,#00H
                MOV     P0,#1FH
                LCALL   CLEAR_LCD
                MOV     A,#82H      ;set cursor to frist position.
                LCALL   CMD
                MOV     A,#'T'
                LCALL   WRITE_TEXT
                MOV     A,#'H'
                LCALL   WRITE_TEXT
                MOV     A,#'A'
                LCALL   WRITE_TEXT
                MOV     A,#'N'
                LCALL   WRITE_TEXT
                MOV     A,#'K'
                LCALL   WRITE_TEXT
                MOV     A,#' '
                LCALL   WRITE_TEXT
                MOV     A,#' '
                LCALL   WRITE_TEXT
                MOV     A,#'Y'
                LCALL   WRITE_TEXT
                MOV     A,#'O'
                LCALL   WRITE_TEXT
                MOV     A,#'U'
                LCALL   WRITE_TEXT
                MOV     A,#0CH
                LCALL   CMD
                LJMP    EXIT
INIT_LCD:       MOV     A,#38H          ;LCD Initialization
                LCALL   CMD             ;2 lines 5x7 matrix
                LCALL   CLEAR_LCD
                MOV     A,#0EH
                LCALL   CMD             ;Display On,Cursor On
```

```
                MOV     A,#06H
                LCALL   CMD         ;Increment Cursor
                RET
CMD:            LCALL   WAIT_LCD
                CLR     RS
                MOV     DATA1,A
                CLR  RW
                SETB EN
                LCALL   DELAY
                CLR  EN
                RET
WAIT_LCD:       SETB    DB7         ;P1.7 AS INPUT
                CLR     RS          ;RS=0
                SETB    RW          ;R/W=1 TO READ
BACK:           CLR     EN          ;E=0
                LCALL   DELAY
                SETB    EN          ;E=1
                JB      DB7,BACK    ;STAY UNTIL BUSY FLAG=0
                RET
DELAY:          MOV     R2,#37H
AGAIN:          MOV     R3,#225H
HERE1:          NOP
                NOP
                DJNZ    R3,HERE1
                DJNZ    R2,AGAIN
                RET
CLEAR_LCD:      MOV     A,#01H      ;Clear LCD Sub-routine
                LCALL   CMD         ;Clear LCD
                RET
WRITE_TEXT:     LCALL   WAIT_LCD
                SETB    RS     ;Data Write On LCD Sub-Routine
                CLR     RW
                MOV     DATA1,A
                SETB    EN
                LCALL   DELAY
                CLR     EN
                RET
UP_DOWN:        NOP
S2:             MOV     A,#'0'
S1:             LCALL   WRITE_TEXT
                LCALL   DELAY
LO1:            JB      UP,G3
                JB      DOWN,G3
                JB      ENT,G3
                SJMP    LO1
G3:             LCALL   DELAY
                LCALL   DELAY
                LCALL   DELAY
                LCALL   DELAY
                LCALL   DELAY
```

```
                LCALL   DELAY
                LCALL   DELAY
                LCALL   DELAY
                LCALL   DELAY
                LCALL   DELAY
                NOP
                JB      UP,IN1
                JB      DOWN,DE1
                JB      ENT,ST1
                SJMP    LO1
IN1:            MOV     R5,A        ;MOV ACC. TO R5
                MOV     A,#10H
                LCALL   CMD         ;mOVE CURSOR LEFT
                MOV     A,R5
                INC     A
                CJNE    A,#3AH,S1
                SJMP    S2
DE1:            MOV     R5,A
                MOV     A,#10H
                LCALL   CMD
                MOV     A,R5
                DEC     A
                CJNE    A,#2FH,S1
                SJMP    S2
ST1:            RET
D_T_H:          ANL     A,#0F0H
                SWAP    A
                MOV     R5,A
                JZ      NOCHANGE
                MOV     A,#00H
AG:             ADD     A,#06H
                DJNZ    R5,AG
                MOV     R6,A
                MOV     A,R4
                CLR     C
                SUBB    A,R6
                RET
NOCHANGE:       MOV     A,R4
                RET
TIMER:          MOV     R7,#14H     ;Timer time=50ms and
                                    ;loop=14H=20D
                MOV     TMOD,#10H   ;Timer 0 mode 1(16 - bit)
REP:            MOV     TL1,#0FEH   ;TL0=4b, Low byte
                MOV     TH1,#4BH
                SETB    TR1         ;Start Timer 0
AGA:            JNB     TF1,AGA
                CLR     TR1         ;Stop  Timer 0
                CLR     TF1    ;Clear Timer 0 flag for next round
                DJNZ    R7,REP      ;Jump for Loop
                RET
```

```
EXIT:       NOP
            SJMP    EXIT
            NOP
            END
```

When system is started the startup message on LCD modules screen appears “PRESS ANY KEY!!!’. When any key is pressed by the user LCD displays the message “TO SET TIME OUT PRESS! ENT!”. “ENT” key press displays the message on the LCD screen as shown below with cursor blink for the first device time out setup.

D1_T= D2_T=
D3_T= D4_T=

Using the UP and Down keys time is set. The set time for each device on the LCD screen look like this,

D1_T=10 D2_T=20
D3_T=30 D4_T=40

Now press “ENT” and then press “RUN”. When the user press RUN button a device control subroutine is executes, and sends the data to the RF module which transmits the data through antenna. The maximum time limit which you can set is 99 seconds. For 00 seconds a particular device is continuously ON for infinite time.

11.6 Component List

For transmitter:

- **SEMICONDUCTORS:**
  IC1 - AT89C51 (Microcontroller)
  IC2 - HT12E (Encoder IC)
  IC3 – 7805 (5V Regulator IC)

- **CAPACITORS**
  C1 – 27 pf
  C2 – 27 pf
  C3 – 10µf, 16V
  C4 – 47µf, 16V
  C5 – 100µf, 25V

- **RESISTORS**
  R1 – 4.7M ohm
  R2 to R10 – 1K ohm
  R11 – 8.2K ohm

- **DIODE**
  D1 to D8, D10 – 1N4148
  D9 - RED LED

- **CONNECTORS**
  J1 – 2 Pin
  J2, J3, J5 – 8 Pin
  CON 1, CON 1-; 4 Pin

- **MISCELLANEOUS**
  RF TRANSMITTER MODULE – TX1
  LCD (J4) – 2 Line and 16 Characters
  S1 – S4: Push to on switches
  X1 – 11.0592 MHz Crystal
  BT 1 – 9 Volt Battery
  IC Sockets – 40 Pin, 16 Pin

For Receiver:

- **SEMICONDUCTORS:**
  U1 – HT12D (DECODER)
  U2 – MN4519B (Multiplexer)
  U3 – ULN2003A (Relay driver IC)
  U4 – 7805 (Voltage Regulator)

- **CAPACITORS**
  C1 – 470nf
  C2 – 100µf, 25V
  C3 – 47µf, 16V

- **RESISTORS**
  R1, R8 – 47k ohm
  R2 to R7 – 1K ohm

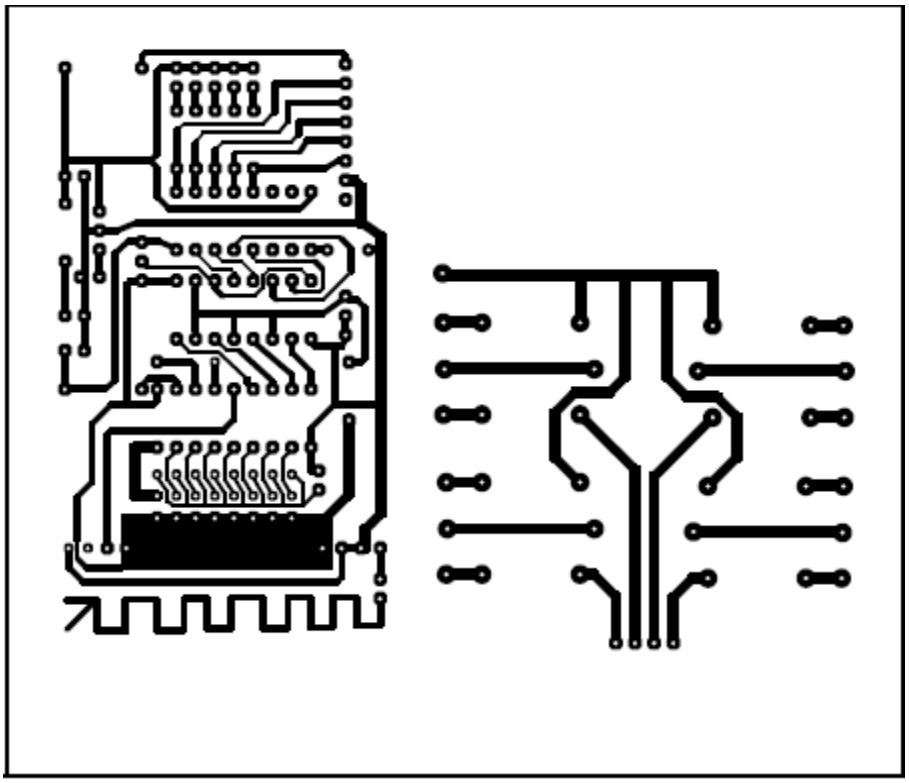
- **DIODE**
  D1 to D8 – 1N4148
  D2 to D6 -- RED LED
  D7 - 1N4007

- **CONNECTORS**
  J1, J7 – 2 Pin
  J2, J3, J4, JP1 – 8 Pin
  J5, J6 – 3 Pin

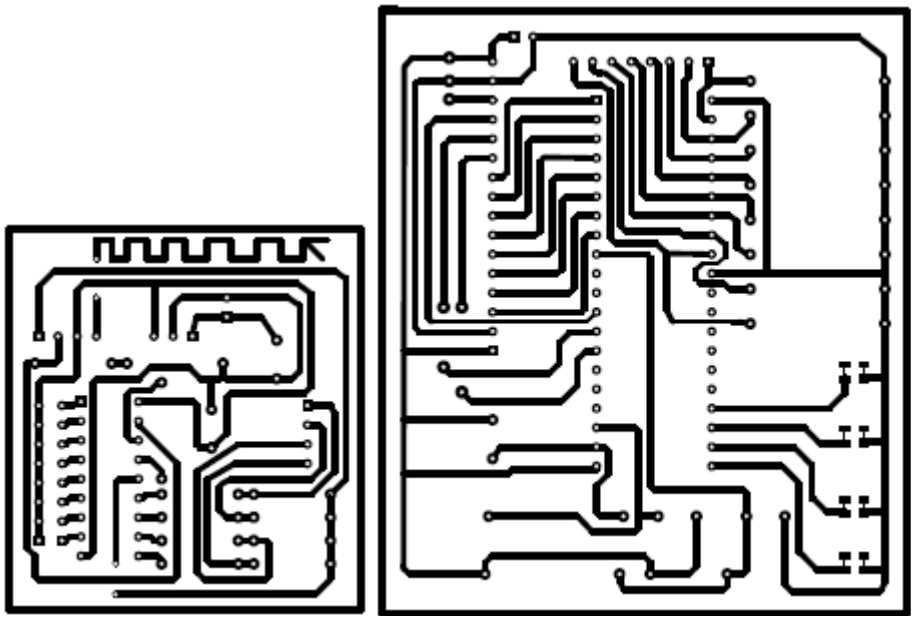
- **MISCELLANEOUS**
  RF RECEIVER MODULE – RX1
  BT 1 – 9 Volt Battery
  IC Sockets – 18 Pin, 16 Pin (2)

Solder Side PCB layout diagram

Figure 11-9 shows the bottom-side PCB layout diagram for the RF Test Kit.



(a) Receiver Section.

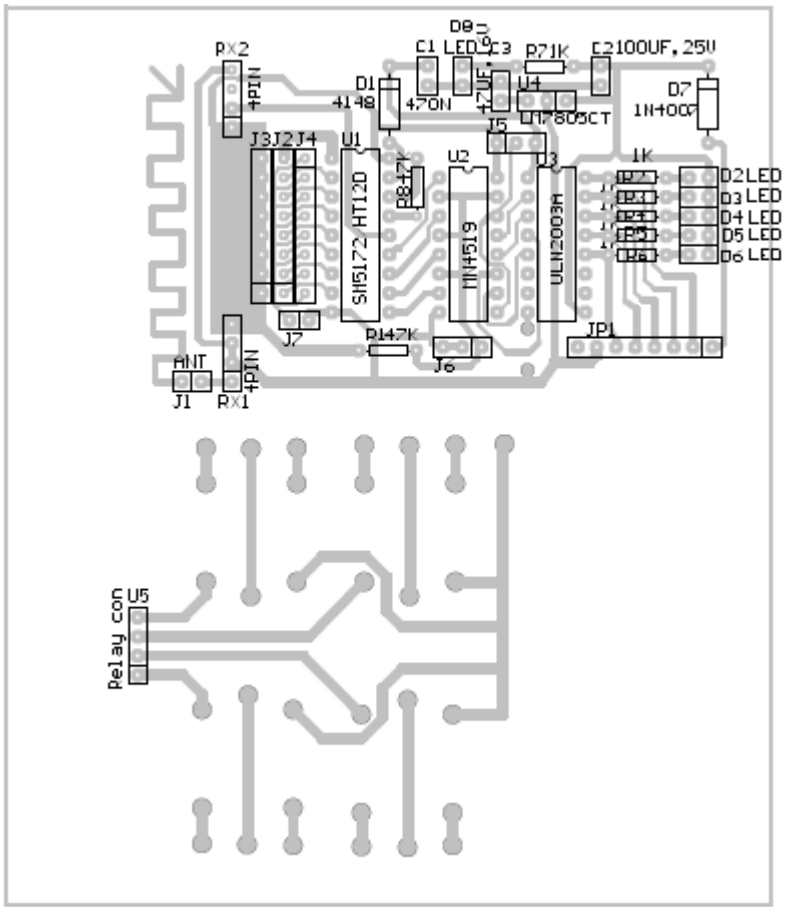


(b) Transmitter Section

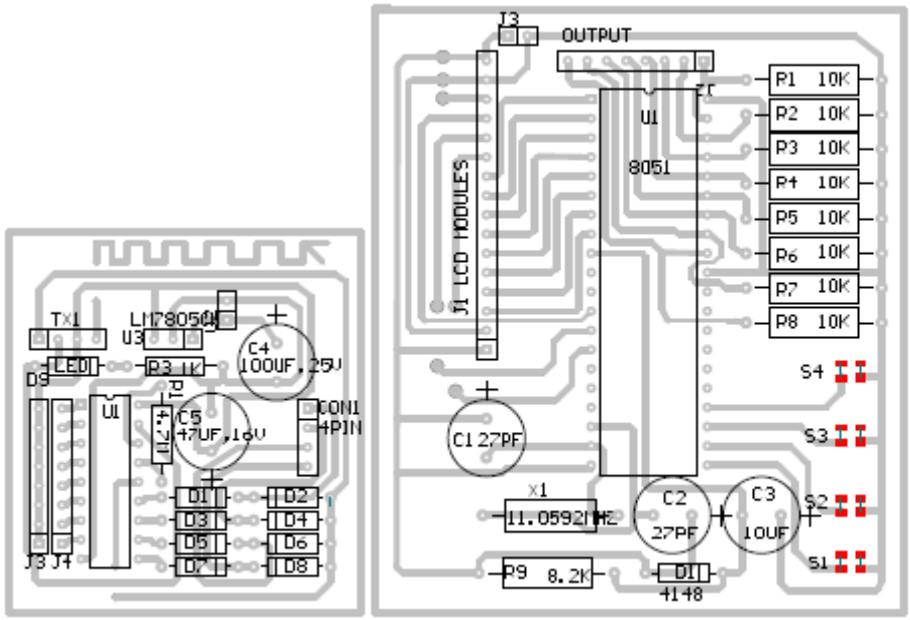
Figure 11-9: Solder Side PCB layout :– (a) Receiver Section
(b) Transmitter Section.

Component layout diagram

Figure 11-10 shows the top-side component layout diagram for the RF Test Kit.



(a) Receiver Section.



(b) Transmitter Section

Figure 11-10: Component layout diagram: – (a) Receiver Section
(b) Transmitter Section.

Figure 11-11 shows the current working picture of the RF test kit in a simple instrument control project. That can be implemented for home automation as well as in the industrial applications.

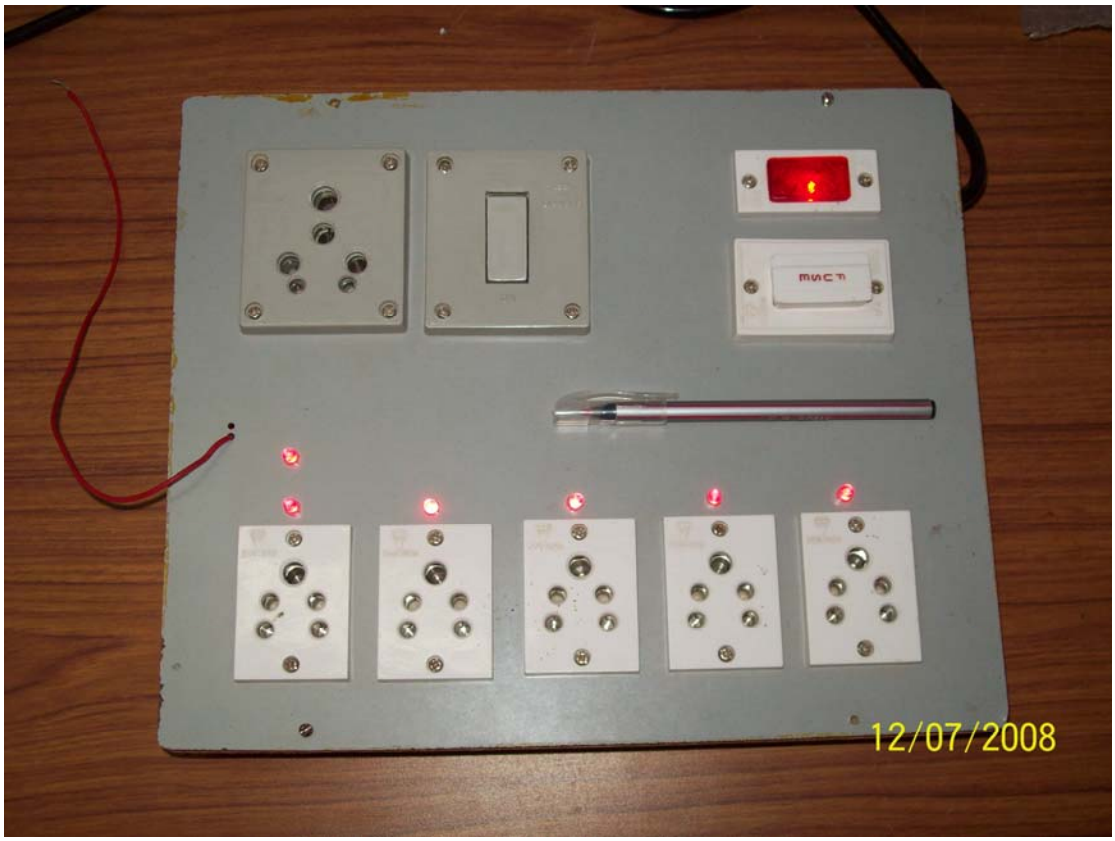


Figure 11-11(a): Receiver Section of the RF Test Kit front end.

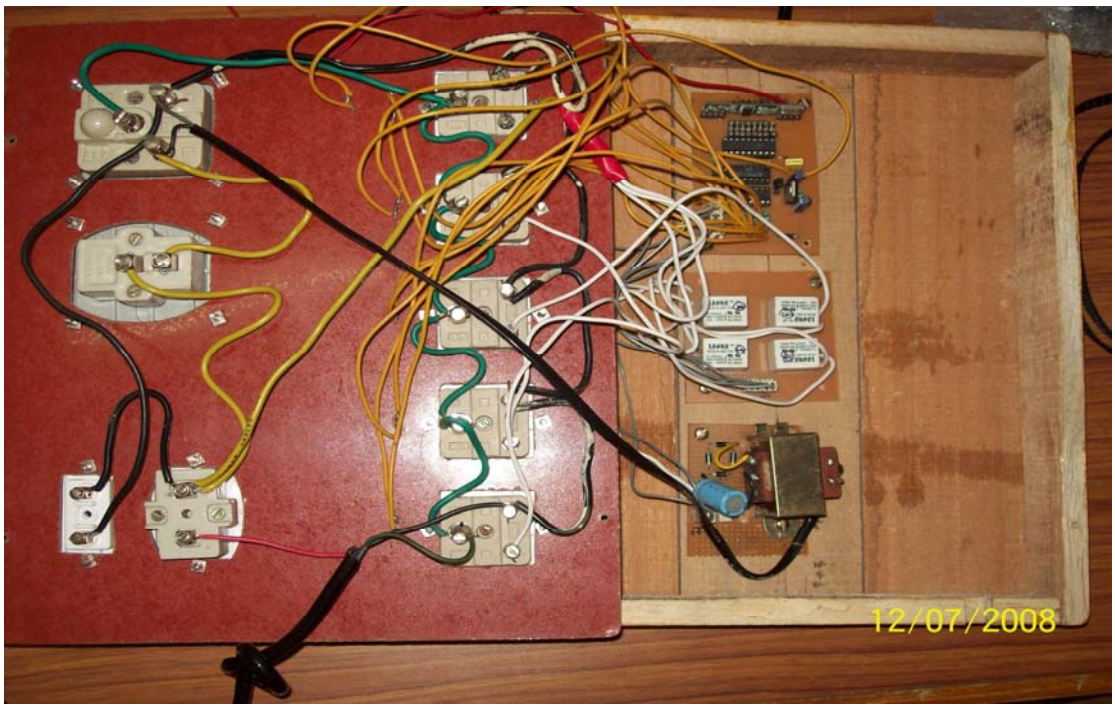


Figure 11-11(b): Inside view of the Receiver Section of the RF Test Kit.

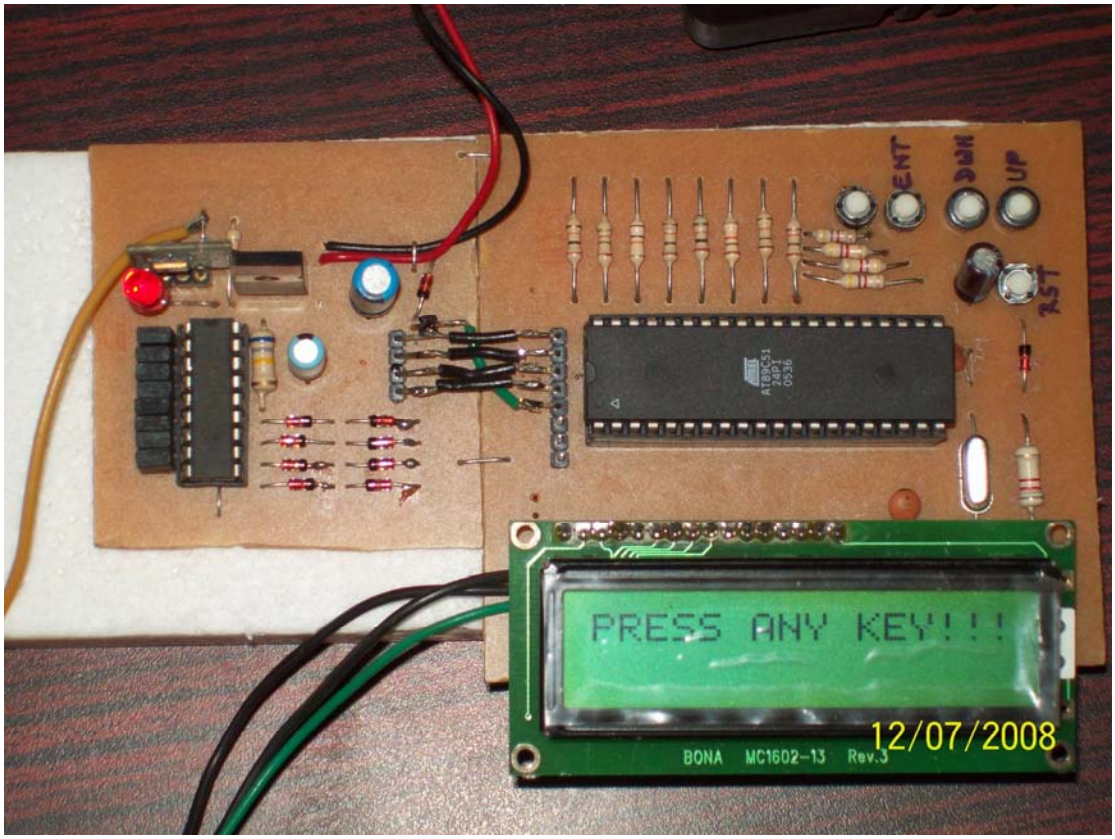


Figure 11-11(C): Transmitter Section of the RF Test Kit.

11.7 Future Developments

Only simplex mode of communication is tested and can be further implemented for the half/full duplex mode by adding transceivers on the both end of the embedded system. More secure and advance RF test kit can be made which can not be hacked by any unauthorized user. It can be done by using encoded RF codes using dedicated microcontrollers on both the ends of the RF test kit system. The RF kit can be implemented on various RF controlled applications. Miniaturization of the RF board is required which takes more space and weight in this case. Layered printed circuit board and SMT components can be used to reduce the size and weight of the overall system.

Chapter 12 ARM7 Based Embedded System: RFID Test Kit and Its Applications

12.1 Introduction

The goal of developing RFID test kit is to explore current RFID and ARM7 based technology and provide documentation for future research projects. RFID can be categorized as low frequency or high frequency and as passive or active. This present work looks mainly at low frequency passive RFID because it is the least expensive and the most common. Low frequency passive RFID works by magnetic induction. A reader device generates an alternating magnetic field from a coil of wire. A tag also has a coil of wire that induces current from the magnetic field. This current provides power to the passive tag. The tag then sends back its own varying magnetic field which is read by the reader device.

RFID module (a reader with ten smart cards) is interfaced with the ARM7 microcontroller to develop a platform for the RFID based applications. In the present work the RFID module is interfaced with the UART port1 of the ARM7 based system, which reads the RFID code embedded in the module and send the same to PC via UART port2. The front-end in the PC is nothing but the windows utility HyperTerminal which communicate with the ARM7 based system serially and displays the tested RDIF cards code read by the system. Figure 12-1 shows the block diagram of the RFID test kit. One of the main advantages of this design is that no external power supply is used all the required power is fed from the PC-SMPS as shown in the block diagram.

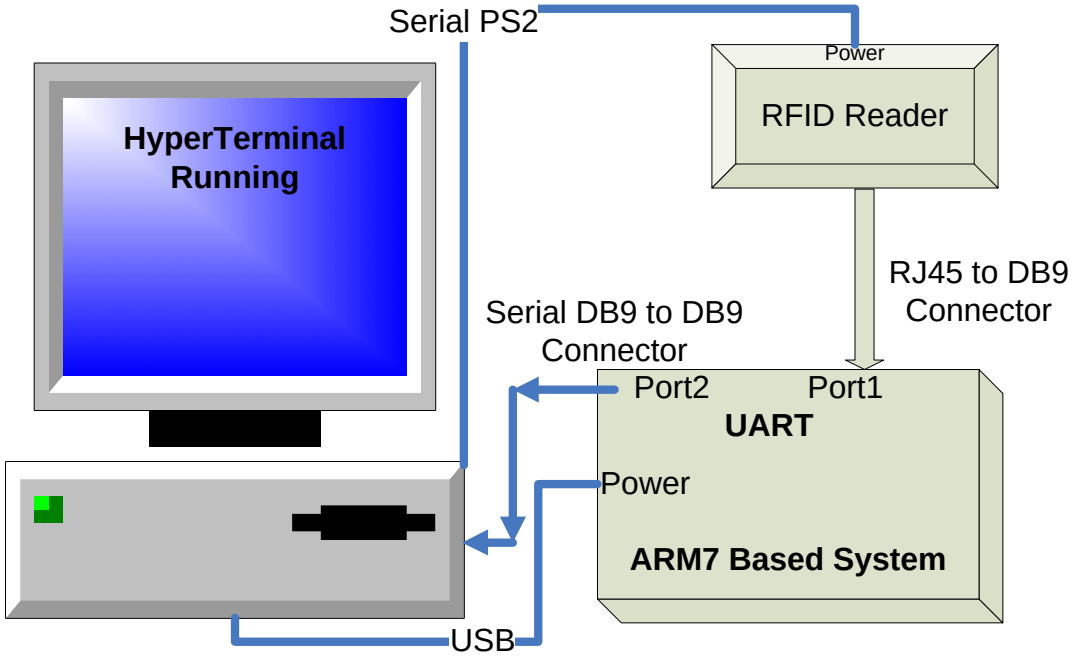


Figure 12-1: Block diagram of the RFID test kit.

12.2 Basics of RFID Test Kit using ARM7 Processor

RFID, short for Radio Frequency Identification, is a technology that enables identification of a tag (that is attached with an entity) by using electromagnetic waves. The function served by RFID is similar to bar code identification, but line of sight signals are not required for operation of RFID. Important components of an RFID system are as listed below.

1. An RFID reader also called transceiver with an antenna and a transceiver.
2. A transponder also called a tag that includes an antenna and a chip.

Table 12-1 shows the differences between the barcode and RFID technologies.

Parameter	Bar Code	RFID
Frequencies used for tag reading	Optical frequencies	Radio frequencies
Type of communication	Line of sight communication	Non-line of sight communication
Data Volume	Physical limitation exists. It is very difficult to read a very long barcode.	Can carry relatively large volume of data.
Range of data readability	Very limited range, less than feet or two.	Can be read up to several feet.
Cost	Cheap	Expensive, but likely to cost less as more industries adopt the technology.

Table 12-1: Comparing Bar code and RFID Technology.

The main advantages of the RFID are listed below:

1. Non-line of sight identification of tags
2. Unattended operations are possible, minimizing human errors and high cost.
3. Ability to identify moving elements that have tags embedded.
4. Larger area of coverage. Up to several feet.
5. Can be used in diverse environments, including live stock, military, and scientific areas.
6. RFID can be used in addition to Bar Code. These two technologies can be complementing each other.
7. Automatic integration with back end software solutions provides end to end integration of data in real time.

The disadvantages of the RFID are listed below:

1. Expensive Compared with bar code.
2. Bulkier, due to embedding of electronic components in the tag. However, with advanced techniques, it is possible to reduce the size, and weight of the tags.
3. Prone to physical/electrical damage.

There are primarily two types of RFID tags. One is active and the other is passive. An active tag is powered using internal battery, where a passive tag gets energized using the power from a tag reader. A passive RFID tag will not have a battery or any kind of power source by itself. It extracts the required energy from a reader. Hence, a passive RFID tag reader must be able to emit stronger electromagnetic signals, and in return, identify very weak signals from the passive RFID tag. Table 12-2 below shows the primary differences between a Passive and Active RFID tags:

	Passive RFID	Active RFID
Power Source	External (Reader provided)	Internal (Battery)
Tag Readability	Only within the area covered by the reader, typically up to 3 meters.	Can provide signals over an extended range, typically up to 100 meters..
Energization	A passive tag is energized only when there is a reader present.	An active tag is always energized.
Magnetic Field Strength	High, since the tag draws power from the electromagnetic field provided by the reader.	Low, since the tag emits signals using internal battery source.
Shelf Life	Very high, ideally does not expire over a life time.	Limited to about 5 years, the life of a battery.
Data storage	Limited data storage, typically 128 bytes.	Can store larger amounts of data.
Cost	Cheap	Expensive
Size	Smaller	Slightly bulky (due to battery)

Table 12-2: Primary difference between passive and active RFID tags.

There are several frequencies that are used for RFID. These include LF, HF, UHF, and Microwave frequencies. The exact frequencies may vary depending on the country where it is used. Table 12-3 shows the frequency range for the RFID devices.

Frequency Range	Description	Typical Applications
<135KHz	Low Frequency, Inductive coupling	Access Control & Security Widgets identification through manufacturing processes Ranch animal identification OEM applications
13.56 MHz	High Frequency, Inductive coupling	Access Control Library books Laundry identification OEM applications
868 to 870 MHz 902 to 928 MHz	Ultra High Frequencies (UHF), Backscatter coupling	Supply chain tracking
2.400 to 2.483 GHz	SHF, Backscatter coupling	Asset tracking Highway toll tags Vehicle tracking

Table 12-3: Frequency range for RFID devices.

In our case low frequency 125 KHz RFID module is used to design RFID test kit. The operation of RFID system can be explained by explaining RFID building blocks, RFID tag block schematic, its operation and backscatter modulation.

RFID building blocks are as explained below:

Tags: A tag is the data carrier and normally contains the ID number, and unique EPC code programmed into the Tag.

Tag Antenna: The tag antenna is connected to the chip in tag. It could be wire or printed using conductive ink.

Reader Antenna: It is a coil included in plastic or similar case, and normally measures 12 -18 inches square.

Reader: A reader captures the data provided by the tag within the detectable area of the Reader. There can be one or more tags within the capture area. A reader is typically capable of reading multiple tags simultaneously.

Savant: This is normally a middleware that interacts with the readers, and communicate with External databases.

A simplified block schematic of RFID tag (also called transponder) is shown in the Figure 12-2 below.

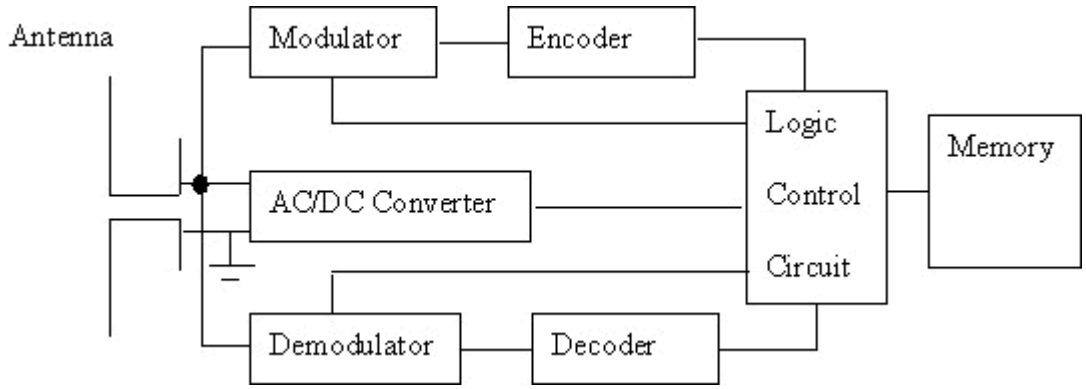


Figure 12-2: Schematic of RFID tag.

The operation of the RFID tag is described below: Handshaking with the Reader (interrogator): The reader continuously emits RF carrier signals, and keeps observing the received RF signals for data.

1. The presence of a tag (for our discussion, we consider only passive tag) modulates the rf field, and the same is detected by the reader.
2. The passive tag absorbs a small portion of the energy emitted by the reader, and starts sending modulated information when sufficient energy is acquired from the rf field generated by the reader. Note that the data modulation (modulation for 0s and 1s) is accomplished by either direct modulation or FSK or Phase modulation.
3. The reader demodulates the signals received from the tag antenna, and decodes the same for further processing.

Backscatter is one of the most widely used modulation schemes for modulating data on to rf carrier. In this method of modulation, the tag coil (load) is shunted depending on the bit sequence received. This in turn modulates the rf carrier amplitude as shown in the Figure 12-3 below. The reader detects the changes in the modulated carrier and recovers the data.

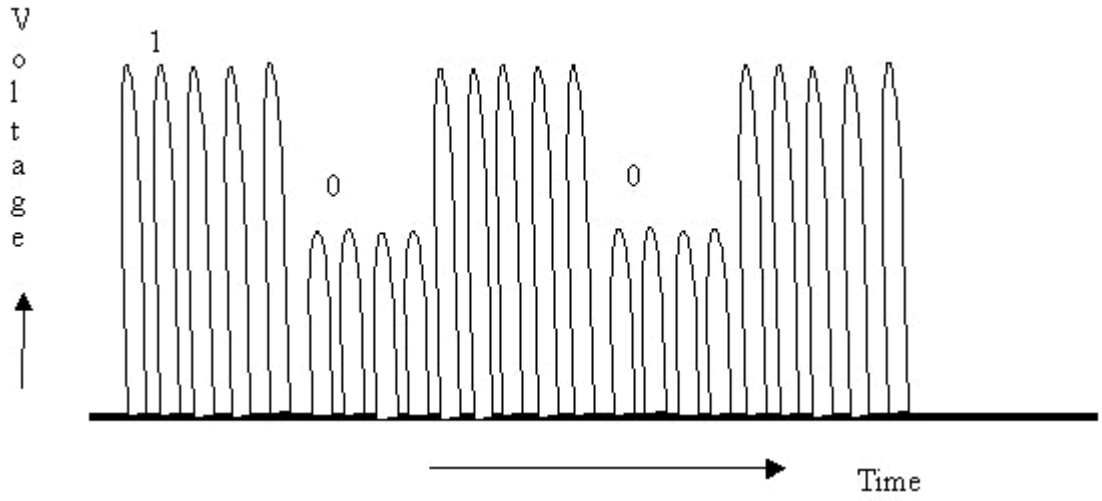


Figure 12-3: Backscatter Modulation.

The above diagram provides a simplified modulated carrier signals from the RFID tag. As seen in the diagram, the encoded binary digits modulate rf carrier. A 1 is represented with high carrier level, and a 0 is represented by a low carrier level (tag coil shunted). The reader demodulates the signals to recover the data, and note that this data is still encoded. The reader decodes the data using suitable decoder, and forwards it for further processing to a computer (or any backend server).

Potential applications for RFID may be identified in virtually every sector of industry, commerce and services where data is to be collected. The attributes of RFID are complimentary to other data capture technologies and thus able to satisfy particular application requirements that cannot be adequately accommodate by alternative technologies. Principal areas of application for RFID that can be currently identified include:

- Transportation and logistics
- Manufacturing and Processing
- Security

A range of miscellaneous applications may also be distinguished, some of which are steadily growing in terms of application numbers. They include:

- Animal tagging
- Waste management
- Time and attendance
- Postal tracking
- Airline baggage reconciliation
- Road toll management

As standards emerge, technology develops still further, and costs reduce considerable growth in terms of application numbers and new areas of application may be expected. Some of the more prominent specific applications include:

- Electronic article surveillance - clothing retail outlets being typical.
- Protection of valuable equipment against theft, unauthorized removal or asset management.
- Controlled access to vehicles, parking areas and fuel facilities - depot facilities being typical.
    - Controlled access of personnel to secure or hazardous locations.
    - Time and attendance - to replace conventional “slot card” time keeping systems.
    - Animal husbandry - for identification in support of individualized feeding programmes.
    - Automatic identification of tools in numerically controlled machines - to facilitate condition monitoring of tools, for use in managing tool usage and minimizing waste due to excessive machine tool wear.
    - Identification of product variants and process control in flexible manufacture systems.
    - Sport time recording.
    - Electronic monitoring of offenders at home.

- Vehicle anti-theft systems and car immobilizer.

A number of factors influence the suitability of RFID for given applications. The application needs must be carefully determined and examined with respect to the attributes that RFID and other data collection technologies can offer. Where RFID is identified as a contender further considerations have to be made in respect of application environment, from an electromagnetic standpoint, standards, and legislation concerning use of frequencies and power levels.

In the current design the RFID module is interfaced with ARM7 based development board as discussed in chapter 06, how to program and proceed with development board LPC21229 is already discussed in that chapter. So only the embedded system development steps are stated here, which can be applicable for any embedded system design.

Before one can start to program embedded systems, he/she needs to learn how to program. While it is not impossible to begin programming using an embedded board, most people find it much easier to learn how to program on a desktop computer before they start to deal with the complexities of cross-compilation, debugging over a JTAG link, etc. Therefore I recommend that one must learn how to program a desktop computer, in C, before he/she can start trying to program an embedded processor. To explain why to recommend learning C is because of the following observations:

- Computers (such as microcontroller, microprocessor or DSP chips) only accept instructions in “machine code” (“object code”).
- Machine code is, by definition, in the language of the computer, rather than that of the programmer. Interpretation of the code by the programmer is difficult and error prone.
- All software, whether in assembly, C, C++, Java or Ada must ultimately be translated into machine code in order to be executed by the computer.
- There is no point in creating 'perfect' source code, if one then make use of a poor translator program (such as an assembler or compiler) and thereby generate executable code that does not operate as one intended.
- When compared to "desktop" processors, embedded processors tend to have limited processor power and very limited memory available: the language used must be efficient.
- To program embedded systems, we need low-level access to the hardware: this means, at least, being able to read from and write to particular memory locations (using 'pointers' or an equivalent mechanism).

Of course, not all of the issues involved in language selection are purely technical:

- No software company remains in business for very long if it generates new code, from scratch, for every project. The language used must support the creation of flexible libraries, making it easy to re-use (well-tested) code components in a range of projects. It must also be

possible to adapt complete code systems to work with a new or updated processor with minimal difficulty.

- Staff members change and existing personnel have limited memory spans. At the same time, systems evolve and processors are updated. Many embedded systems have a long lifespan. During this time, their code will often have to be maintained. Good code must therefore be easy to understand now, and in five years time (and not just by those who first wrote it).
- The language chosen should be in common use. This will ensure that you can continue to recruit experienced developers who have knowledge of the language. It will also mean that your existing developers will have access to sources of information (such as books, training courses, WWW sites) which give examples of good design and programming practice.

Even this short list immediately raises the paradox of programming language selection. From one point of view, only machine code is safe, since every other language involves a translator, and any code you create is only as safe as the code written by the manufacturers of the translator. On the other hand, real code needs to be maintained and re-used in new projects, possibly on different hardware: few people would argue that machine code is easy to understand, debug or to port.
Inevitably, therefore, one needs to make compromises; there is no perfect solution. All we can really say is that we require a language that is efficient, high-level, gives low-level access to hardware, and is well defined. In addition - of course - the language must be available for the platforms we wish to use. Against all of these points, C scores well.
One can summarize C's features as follows:

- It is a 'mid-level' language, with 'high-level' features (such as support for functions and modules), and 'low-level' features (such as good access to hardware via pointers);
- It is very efficient;
- It is popular and well understood;
- Even desktop developers who have used only Java or C++ can soon understand C syntax;
- Good, well-proven compilers are available for every embedded processor (8-bit to 32-bit or more);
- Experienced staff are available;
- Books, training courses, code samples and WWW sites discussing the use of the language are all widely available.

Overall, C's strengths for embedded system development greatly outweigh its weakness. It may not be an ideal language for developing embedded systems, but is unlikely that a perfect language will ever be created.

When one is familiar with desktop C, I suggest that he/she should begin to explore embedded systems using a simple processor. For example, the 8051 microcontroller is a popular "starter" processor which is still used in many "real" systems as I did in my earlier studies.

Try to implement the small programs on the low cost development boards available in the market or make your own as I did in the chapter 06. Download keil compiler from the www.keil.com and start experimenting with the flashing LED code, a switch interface, an RS-232 interface, and learn more about FPGA/CPLD devices to make the system on chip.

12.3 Circuit Diagram and Connection details

The circuit diagram of the RFID test kit is divided into two sections:

1. LPC2129 ARM7 Development Board as shown in the Figure 12-4.
2. RFID module interface serially with the LPC2129 board.

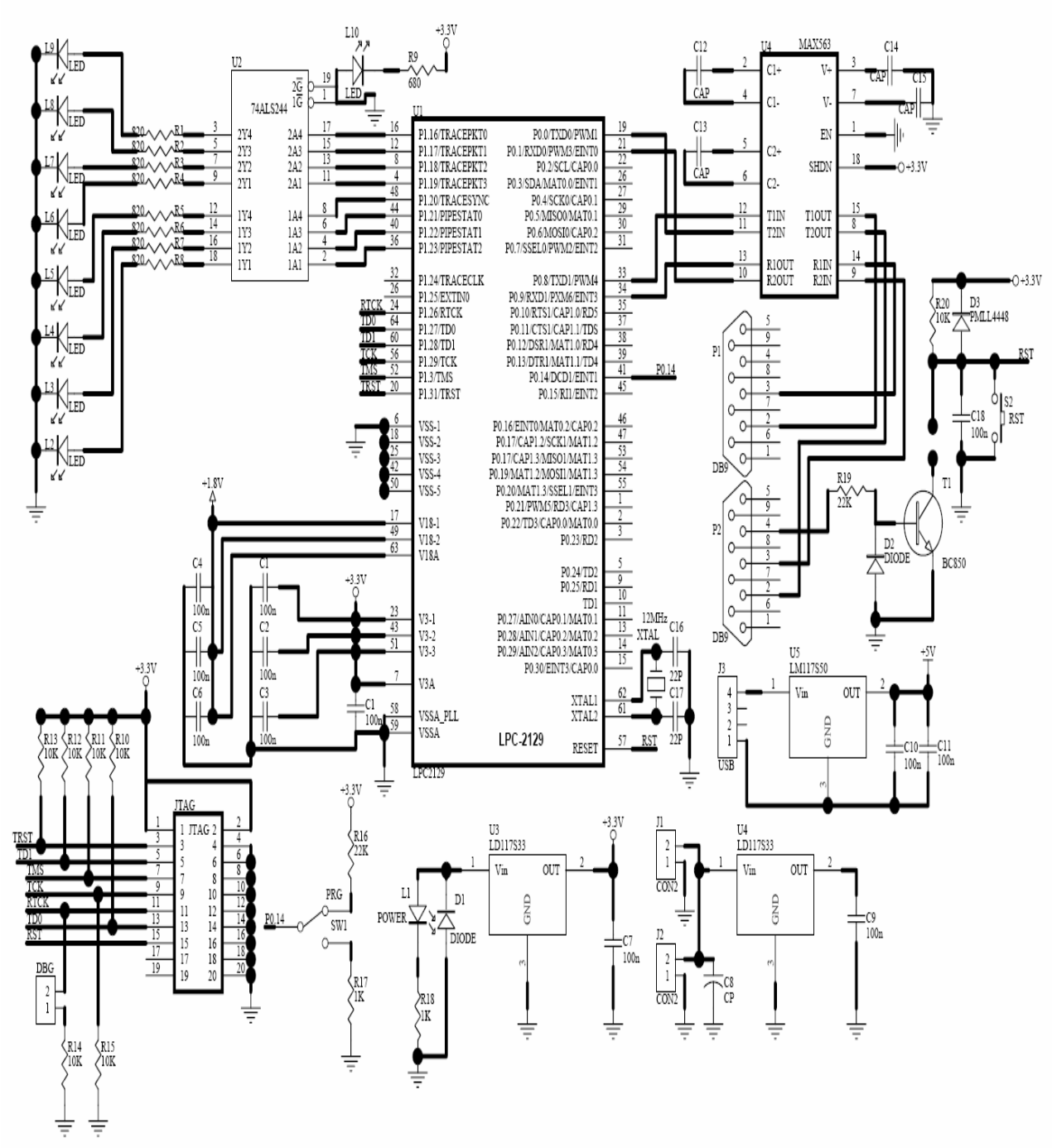


Figure 12-4: LPC2129 ARM7 Development Board.

Figure 12-5 shows the RFID module: RFID reader and one of the cards with an ID written on it. The internal block schematic is already shown in the Figure 12.2 earlier.

Figure 12-5: RFID module used in the RFID test kit.

The connections of the LPC2129 board and the RFID module are as given below:

1. Power supply for ARM board through USB cable from PC.
2. Connect the RFID cable to RFID module (RJ45 end) as shown in the Figure 12.5 above.
3. Connect the other end of RFID cable to P1 of ARM board through male-to-male serial cable.
4. Remove PS2 mouse cable and reconnect through the PS2 port of RFID cable.
5. Connect the male-to-female serial cable from PC to P2 of ARM board.

Once the connections are done dump the RFID.hex file in the LPC2129 board using Philips flash utility software as discussed in chapter 06. The software development and algorithm is provided in the next coming sections.

12.4 Software Flowchart

Figure 12.6 shows the software flowchart of the RFID test kit. The program is running in the ARM7 processor with infinite loop.

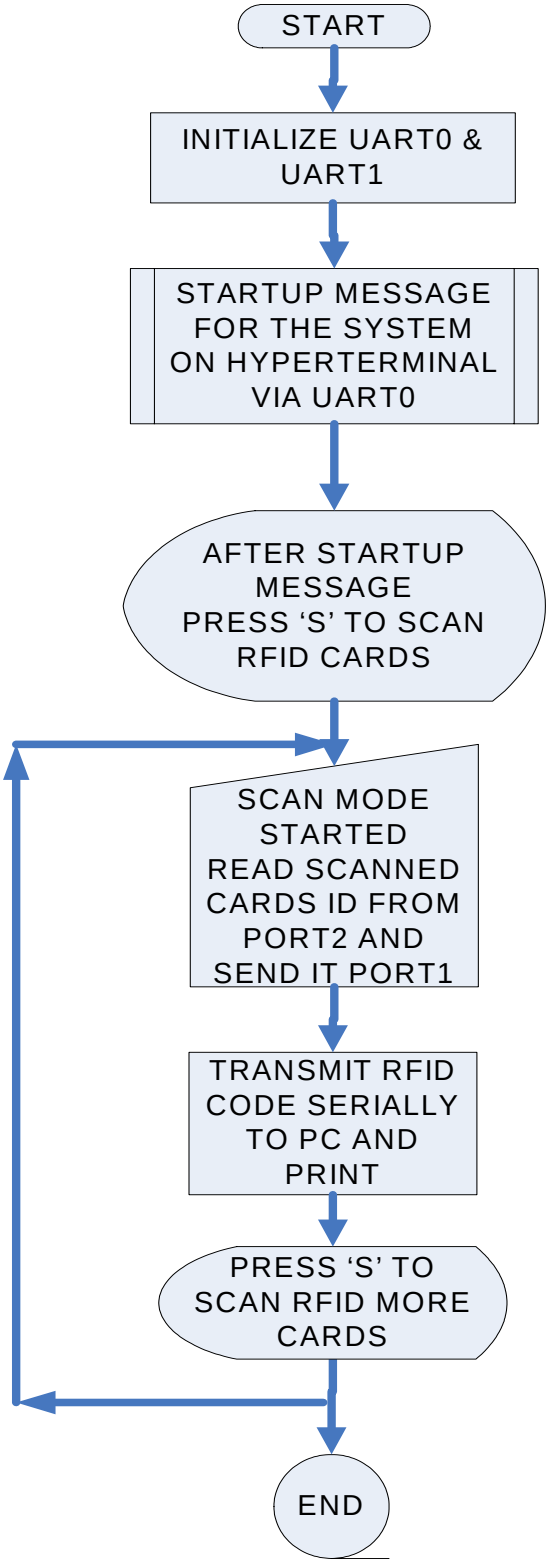


Figure 12-6: Software Flowchart for the RFID Test Kit.

12.5 Software Section

The software for the RFID test kit is written in Embedded C and complied using keilUv3 compiler and output of the compiler (RFID.hex) is programmed into the LPC2129 using Philips flash utility software. All the development tools are freely available from the internet.

The algorithm for the RFID test kit is written below:

1. Enable UART0 and UART1 Interrupt
2. Set UART1 & UART0 as IRQ
3. Set SLOT0 and SLOT1 for UART0 and UART1 interrupt
4. Set Interrupt Vector
5. Initialize serial communication
6. Print the startup message on the HyperTerminal
7. If RFID Card is scanned from the module, read the cards ID using UART1 and send the ID to UART0.
8. Display the cards ID.
9. Prompt user to continue scanning the cards by key press ‘S’.
10. Provide proper threading for wrong character pressed through out the program.

The source code of the RFID test kit is given below and is self explanatory.

```
#include "lpc21xx.h"
#include "string.h"


unsigned char flagp1=0;
volatile unsigned int i;
char RxU0_Data (void);
char RxU1_Data (void);
void init_serial (void);
void UARTINT1(void) __irq;
void UARTINT0(void) __irq;
void delay(void);
char TxU0_Data (char data);
char TxU1_Data (char data);
void print_serial(const char *str);
void Add_mode(void);
unsigned char temp_add;

typedef union { volatile unsigned char array[14];      // ID data base declaration
                                        volatile long long var;

                                        }rfid_object;
volatile rfid_object add_id[50];
volatile rfid_object temp_id;
```

```
/*------------ Read ID and compare the ID subroutine----------*/

 void ID_read(void)
  {
   unsigned int j;
   unsigned int k;
   unsigned char flag = 0;
   print_serial("READ MODE.....");
   TxU0_Data(0x0a);
                    for (j=0; j<14; j++)
                    {
          temp_id.array[j]= RxU1_Data ();
                     TxU0_Data(temp_id.array[j]);
                      }
                   TxU0_Data(0x0a);
                   for (k=0;k<50;k++)
                                        {

                                  if(add_id[k].var==temp_id.var )
                                    {

                                          flag++;
                                  }

                                  }
                                  if(flag!=0)
                                  {
                                  print_serial("Success..... ");
                                  TxU0_Data(0x0a);


                                  }
                                  else
                                  {
                                          print_serial("Failure..... ");
                                          TxU0_Data(0x0a);

                                  }

  }
```

```
 /*----------- Initialisation of UART0 and UART1-----------------*/

void init_serial (void)
{

        PINSEL0 = 0x00050005;   // Enable RxD0/1 and TxD0/1
        U0LCR = 0x83;           // 8 bits, no Parity, 1 Stop bit
        U0DLL = 97;             // 9600 Baud Rate @ 12MHz VPB Clock
        U0LCR = 0x03;           // DLAB = 0
        U0IER = 0x01;           // Enable Transmit interrupt and
                                // Recieve interrupts
        U1LCR = 0x83;           // 8 bits, no Parity, 1 Stop bit
        U1DLL = 97;             // 9600 Baud Rate @ 12MHz VPB Clock
        U1LCR = 0x03;           // DLAB = 0
        U1IER = 0x01;           // Enable Transmit interrupt and
                                // Recieve interrupts

}

char TxU0_Data(char data)
{

        while (!(U0LSR & 0x20));    // Check whether UART0 is busy sending
                                        any
                                // data
   U0THR = data;                // Once free Send data
   return (U0THR);

}


char RxU0_Data (void)
{

        while (!(U0LSR & 0x01));  // Check if UART0 is busy getting data from
                                  // serial port

        return (U0RBR);           // Return the received data

}

char TxU1_Data (char data)
{
          U1THR = data;
        while (!(U1LSR & 0x20));  // Check whether UART1 is busy sending
any
                                  // data
                                  // Once free Send data
        return (U1THR);
}
```

```
char RxU1_Data (void)
{

        while (!(U1LSR & 0x01));   // Check if UART1 is busy getting data from
                                   // serial port

        return (U1RBR);            // Return the received data

}

 void print_serial(const char *str)   // Function for printting string in serial port
{
        unsigned char l,len;
        len = strlen(str);
        for (l=0;l<len;l++)
                {
                TxU0_Data ((*str));

                 str++;
                }
                }
```

/* MAIN FUNCTION*/

```
int main()
{
        char char1;
  //IO1DIR = 0x00ff0000;
        VICIntEnable  |= 0xc0;       // Enable UART1 &UART0 Interrupt
        VICIntSelect = 0x00;         // Set UART1 &UART0 as IRQ
        VICVectCntl0 = 0x26;         // Set SLOT0 for UART0 Interrupt

        VICVectCntl1 =0x27;          // Set SLOT1 for UART1 Interrupt
        VICVectAddr0  = (unsigned int)UARTINT0;       // Set interrupt vector
        VICVectAddr1  = (unsigned int)UARTINT1;       // Set interrupt vector


        init_serial();               // Initialize Serial communication
        TxU0_Data(0x0d);
        TxU0_Data(0x0d);

print_serial("*****RFID Test Kit*****");
        TxU0_Data(0x0d);

print_serial("Department of Electronics, Saurashtra University, Rajkot");
        TxU0_Data(0x0d);
        TxU0_Data(0x0d);

print_serial("+++++Developed by Mr. AnandKumar Atalbihari Bhaskar++++");
```

```
again:
        TxU0_Data(0x0d);
        TxU0_Data(0x0d);

print_serial("Press S to scan and read the RFID CARDs ID");
moreid:
        TxU0_Data(0x0d);
        char1 = RxU0_Data();
        if (char1=='s'|| char1=='S')

//Read the data from port2 and send to port1 which connected to PC serial
//port and save it microcontroller memory.
        {
        unsigned int j;
        U1IER = 0x00;
print_serial("SCAN MODE STARTED.....");
        TxU0_Data(0x0d);
        print_serial("RFID CARDs ID = ");
                for (j=0; j<14; j++)
                {
  // add_id[i].array[j] = RxU1_Data ();
   //TxU0_Data(add_id[i].array[j]);
   TxU0_Data(RxU1_Data ());
        }

TxU0_Data ('e');

//i++;
//U1IER = 0x01;
        TxU0_Data(0x0d);
        TxU0_Data(0x0d);
print_serial("To Scan more IDs Press S again!");
        TxU0_Data(0x0d);
        TxU0_Data(0x0d);
        goto moreid;
        }
        else
                print_serial("Wrong Parameter entered!!!");
                goto again;

}
   /*----------- ISR for UART0------*/
void UARTINT0(void) __irq
{

volatile unsigned     int p;            // IRQ routine for UART1 interrupt
unsigned temp;
         VICVectAddr = 0x00000000;
         VICIntEnable  &= ~((unsigned long int) 0xc0);
```

```
          p= VICIRQStatus;          // Reading the status flags to clear the flags
          p = U0IIR;
                U0IER = 0x00;

         temp =         U0RBR;       //RxU0_Data ();
           if(temp=='A')              // checking for ADD mode
                  {
                  U1IER = 0x00;
                  VICIntEnable  &= ~((unsigned long int) 0x40);
                   //Add_mode();
                  VICIntEnable  = 0xc0 ;

                   TxU0_Data(0x0a);
                   U1IER = 0x01;
                  }
           else if(temp=='R')          // checking for Read mode
           {

             flagp1 = 1;

           }
                 U0IER = 0x01;

}




 /*---------- ISR for UART1------*/

void UARTINT1(void) __irq
{

// IRQ routine for UART1 interrupt
volatile unsigned     int p;

          VICVectAddr = 0x00000000;
          VICIntEnable  &= ~((unsigned long int) 0xc0);

          p= VICIRQStatus;            // Reading the status flags to clear the flags
          p = U0IIR;
          U1IER = 0x00;

                   if(flagp1==1)
     {
       ID_read();                     // reading the ID and Comparing
        }
 U1IER = 0x01;
}
```

Working of the RFID Test Kit:

Initially when the RFID Test Kit is connected to the PC and power is provided; the startup message on the HyperTerminal (In present work I had used SPJTerm.exe to receive data serially from the RFID kit) will appear and is shown in Figure 12-7.

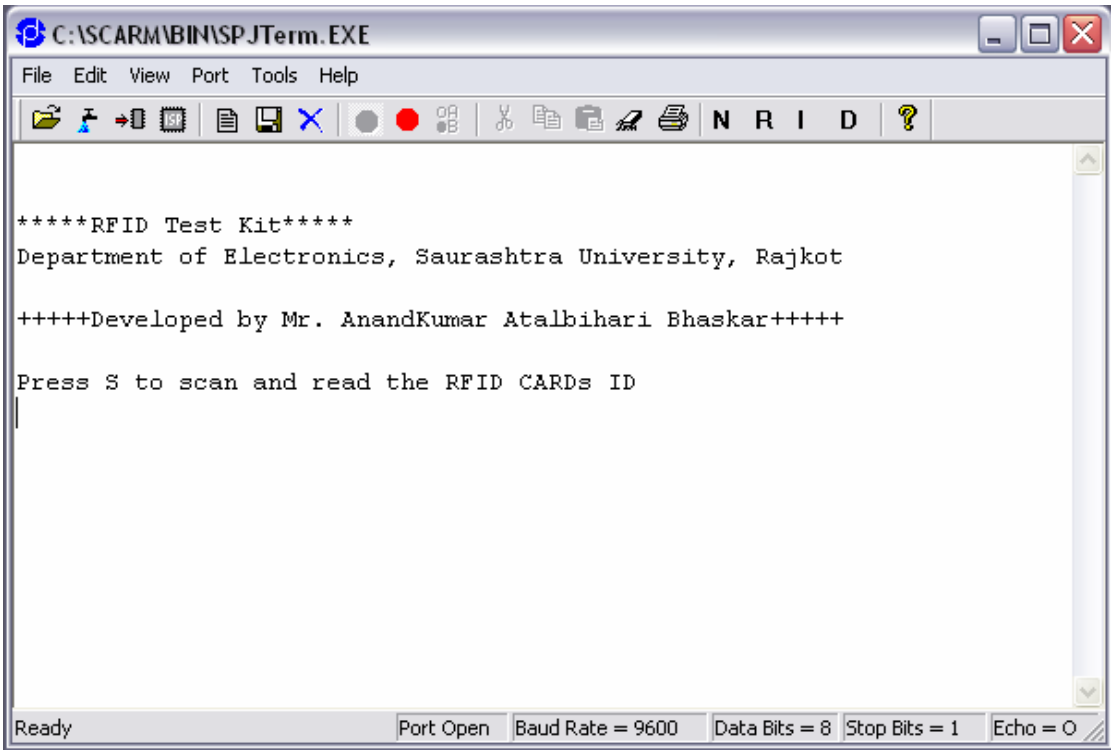


Figure 12-7: Startup message for the RFID Test Kit.

When the user wants to scan and read the RFID cards ID he/she has to press ‘S’ button from the keyboard, other wise the system will not display the scanned cards ID. If any wrong character is entered by the user the string “Wrong Parameter is entered!!!” will be displayed. If the user press the correct character ‘S’ to scan and read cards ID scan mode will start as shown in the Figure 12-8 below.

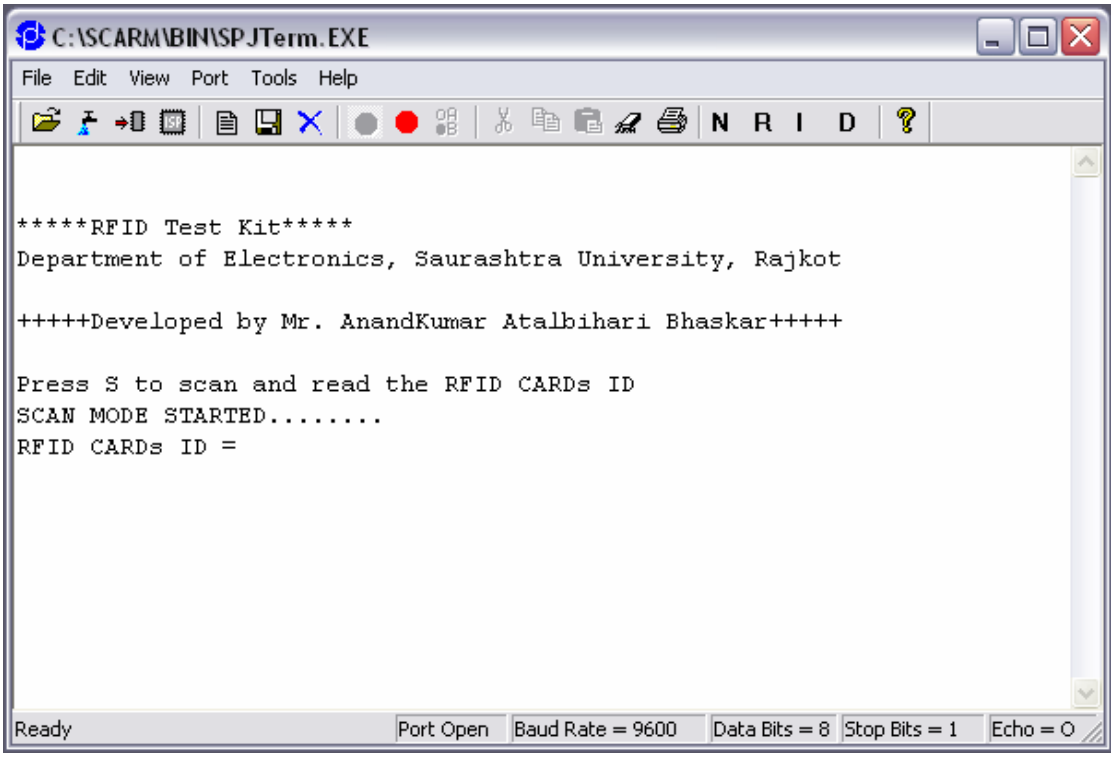


Figure 12-8: Scan mode Started: Scan the card from the RFID module.

Once the scan mode is started the system is now ready to receive the cards ID. So whenever after this card is scanned by the module it will display the cards ID once and allows the user to scan more ID's by pressing 'S' again as shown in the Figure 12-9 below.

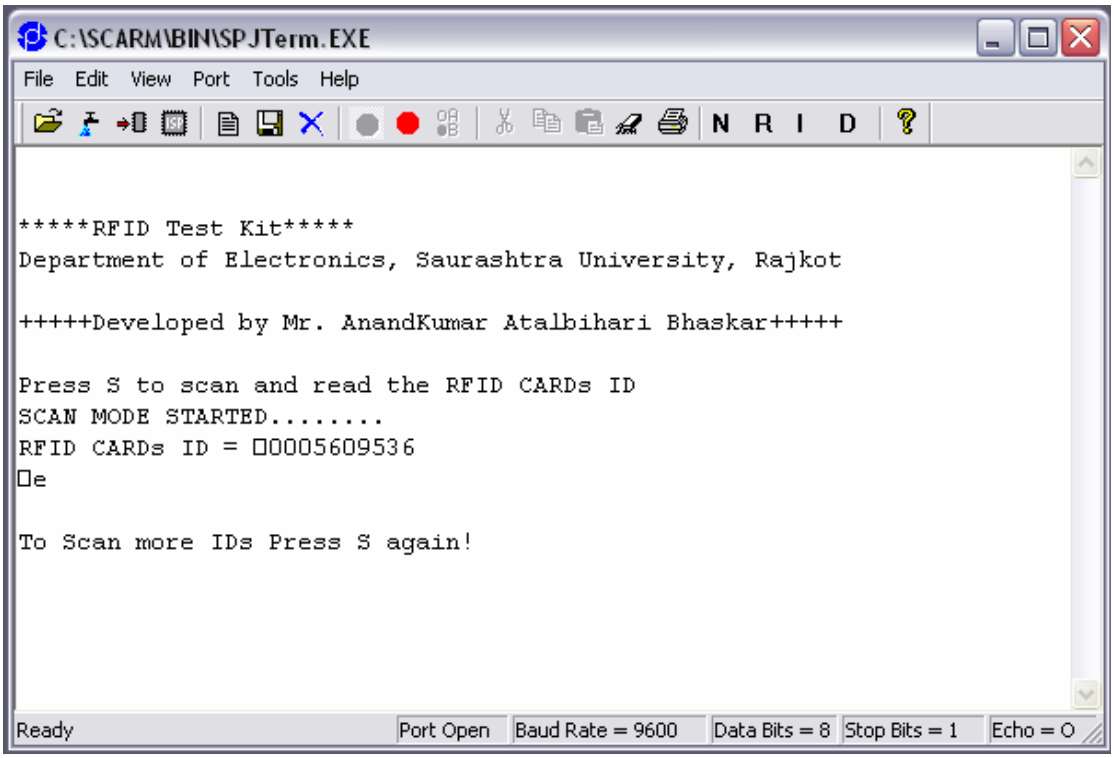


Figure 12-9: Cards ID read by the RFID Test Kit Successfully.

12.6 Component List

1. RFID module with ten cards.
2. LPC2129 prototype board with two UART.
3. DB9 connectors and serial cables.
4. RJ45 to USB cable.
5. KeilUV3 software.
6. Philips Flash Utility Software.
7. HyperTerminal program running on windows environment.
8. Personal Computer with PS2 mouse port, USB port and RS232 Com Port.

12.7 Future Developments

The present work is in the form of test kit which can be applied as an access control system or as daily inventory system for attendance by doing major changes in the software and making dedicated front end for the system supporting large data base. By doing this work a better understanding of Embedded C for ARM based design is developed and lots of things can be done using ARM core of microcontrollers.
An RFID enabled car parking is a system which can be one of the project that can be developed by using RFID technology as discussed in the current chapter.

Chapter 13 Conclusion and Future Research Direction

The thesis title “DESIGN AND DEVELOPMENT OF MICROCONTROLLER SYSTEM, EMBEDDED CIRCUITS AND APPLICATIONS” is fully demystified in the chapters written to support it. In above twelve chapters mostly all of the objectives are successfully achieved. All the experimentation for achieving the desired goals is performed in the Department of Electronics, Saurashtra University, Rajkot research lab under guidance of Dr. H. N. Pandya.

With the relevance to the current thesis work I had published few research papers in technical journals, magazines and seminars. The detailed list of that is presented in the Appendix F.

For future studies I had a list of Embedded System projects as listed below to be continued with my extra effort in the research field.

1. Digital dash board for vehicles-tracking police man using RFID.

Objective:
This is an advanced technology used to track the vehicle number & registration details using contact less rfid tags. Some times people manipulate the original details like vehicle number, registration details to the police man, where police finds difficult to find the culprit. So to eliminate this problem, in this project we are using a contact less rfid tags which are placed at the wheels of the vehicle. There is an rfid reader placed under the platform through which the vehicle passes. From reader the information is passed to the microcontroller through RS232 communication, where controller decodes the information using the predefined database for the respective rfid tags. Every rfid tag has a unique code which is transmitted to the reader through wireless. There is no power or battery supply given to the rfid tags. This decoded information is displayed with vehicle number & registration details at the dash board using the lcd display, so that one can know the original data with more security.

2. Microcontroller based remote notice board with RS232 communication.

Objective:
The device is used to communicate the data through RS232 communication. At transmitter side there is a pc where the user can enter the data to be transmitted to the receiver side. From pc the data is communicated through serial communication (RS232) to the microcontroller from which the data is decoded. Through lcd display interface to the microcontroller the data is displayed. It has a unique address code for transmitter so that it can match to particular receiver. It consumes less power, which is of Compact & Cost effective solution.

3. RFID based lift control system using microcontroller

Objective:
This is an advanced system for lift control system using contact less RFID cards. Each rfid card has a unique id number with read only option. So that it can be accessed whenever it is required using the rfid reader with coil sensing unit. This system enables the door to the building whenever the valid rfid card is inserted to the device. Only a valid person can access the door. It also has a feature like time period for opening & closing the lift door, so that device can give the permission for access of the door for certain period as predefined in the microcontroller program. This way the system is reliable compared to the analog devices with more security. Microcontroller communicates with the rfid reader through serial communication (RS232) Using USART, MAX232. It decodes the address of the rfid card & checks whether the card is valid or invalid. Based on this the user is able to access the lift which is controlled by the relay unit. In this way we can avoid the unauthorized persons to access the lift & it can also save the power.

4. Attendance monitoring system

Objective:
Latest and Advanced Software product SMART CARD SYSTEM which can be used in IT companies, Banks, Industries, office, institutions etc.

- This device as the name implies automates the attendance monitoring system.
- Unlike the conventional magnetic tapes, which are prone to wear and tear, this device makes use of smart card technology.
- Smart card comes with inbuilt memory in which personal details like name, employee ID, division and other details can be stored.

5. Microcontroller based industrial equipment control system at remote location using pc interface

Objective:
Application of this project is to communicate with the pc to **Microcontroller** through serial communication using USART & Max232. Now a day's many devices are developed to communicate with the pc interface, so that it can be user friendly to interact with the devices. In this application microcontroller will operate the external devices using pc interface where the user sets the command to operate the particular device using application software at the pc. Where the pc sends the information to the **Microcontroller** through serial communication using USART & Max232 modules. By decoding the information microcontroller will operate the specified device which set by the user through pc.

6. Microcontroller based bus location announcement system using RFID

Objective:

Object locator can be done in two ways that is using GPS based or RFID (TX & RX) based. If user wants to find the location of the bus through out the globe then he/she should need to go with the global positioning system (GPS). Since global positioning system is costly one can shift to RF to locate the bus at a specified area through the RF communications. Which is a low cost solution unlike to the GPS? From RF transmitter the data is communicated to the microcontroller through serial communications which will intern shows the location address which is predefined in the controller for the particular location, where as in RF communication it is restricted to a specified area. Whenever object moves in that specified area RFID transmitter sends the location address to the receiver which is attached to the respective bus stop, which will communicate with the controller to decode the address of the location. Then the controller enables the voice chip for the particular address location. Through the voice chip it is interfaced to the speaker to announce the address of that particular location at that bus stop.

There are many other good projects also, which can be done successfully in the department but would not like to disclose all of them at this time.

APPENDIX

Appendix A: The 8051 Instruction Set

Appendix B: ARM7 Thumb Instruction Set

Appendix C: ASCII Codes

Appendix D: Data Sheets

Appendix E: References

Appendix F: Paper Published

Appendix A: The 8051 Instruction Set

Here one can find complete instruction set of 8051 micro-controller. Complete information regarding each instruction like operational explanation, addressing mode, no. of byte occupied, no. of cycles used etc is given. So just, go through it. It's a ready reference.

All 8051 instructions are broadly classified in to four groups data moving, logical, arithmetic and branching.

Data moving / handling Instructions: -

Mnemonics	Operational description	Addressing mode	No. of bytes occupied	No. of cycles used
Mov a,#num	Copy the immediate data num in to acc	immediate	2	1
Mov Rx,a	Copy the data from acc to Rx	register	1	1
Mov a,Rx	Copy the data from Rx to acc	register	1	1
Mov Rx,#num	Copy the immediate data num in to Rx	immediate	2	1
Mov a,add	Copy the data from direct address add to acc	direct	2	1
Mov add,a	Copy the data from acc to direct address add	direct	2	1
Mov add,#num	Copy the immediate data num in to direct address	direct	3	2
Mov add1,add2	Copy the data from add2 to add1	direct	3	2
Mov Rx,add	Copy the data from direct address add to Rx	direct	2	2
Mov add,Rx	Copy the data from Rx to direct address add	direct	2	2
Mov @Rp,a	Copy the data in acc to address in Rp	Indirect	1	1
Mov a,@Rp	Copy the data that is at address in Rp to acc	Indirect	1	1
Mov add,@Rp	Copy the data that is at address in Rp to add	Indirect	2	2

Mov @Rp,add	Copy the data in add to address in Rp	Indirect	2	2
Mov @Rp,#num	Copy the immediate byte num to the address in Rp	Indirect	2	1
Movx a,@Rp	Copy the content of external add in Rp to acc	Indirect	1	2
Movx a,@DPTR	Copy the content of external add in DPTR to acc	Indirect	1	2
Movx @Rp,a	Copy the content of acc to the external add in Rp	Indirect	1	2
Movx @DPTR,a	Copy the content of acc to the external add in DPTR	Indirect	1	2
Movc a,@a+DPTR	The address is formed by adding acc and DPTR and its content is copied to acc	indirect	1	2
Movc a, @a+PC	The address is formed by adding acc and PC and its content is copied to acc	indirect	1	2
Push add	Increment SP and copy the data from source add to internal RAM address contained in SP	Direct	2	2
Pop add	copy the data from internal RAM address contained in SP to destination add and decrement SP	direct	2	2
Xch a, Rx	Exchange the data between acc and Rx	Register	1	1
Xch a, add	Exchange the data between acc and given add	Direct	2	1
Xch a,@Rp	Exchange the data between acc and address in Rp	Indirect	1	1
Xchd a, @Rp	Exchange only lower nibble of acc and address in Rp	indirect	1	1

Logical Instructions: -

Mnemonics	Operational description	Addressing mode	No. of bytes occupied	No. of cycles used
Anl a, #num	AND each bit of acc with same bit of immediate num, stores result in acc	Immediate	2	1
Anl a, add	AND each bit of acc with same bit of content in add, stores result in acc	Direct	2	1
Anl a, Rx	AND each bit of acc with same bit of content of Rx, stores result in acc	Register	1	1
Anl a, @Rp	AND each bit of acc with same bit of content of add given by Rp, stores result in acc	Indirect	1	1
Anl add, a	AND each bit of acc with same bit of direct add num, stores result in add	Direct	2	1
Anl add, #num	AND each bit of direct add with same bit of immediate num, stores result in add	direct	3	2
orl a, #num	OR each bit of acc with same bit of immediate num, stores result in acc	Immediate	2	1
orl a, add	OR each bit of acc with same bit of content in add, stores result in acc	Direct	2	1
orl a, Rx	OR each bit of acc with same bit of content of Rx, stores result in acc	Register	1	1
orl a, @Rp	OR each bit of acc with same bit of content of add given by Rp, stores result in acc	Indirect	1	1
orl add, a	OR each bit of acc with same bit of direct add num, result in add	Direct	2	1

orl add, #num	OR each bit of direct add with same bit of immediate num, stores result in add	direct	3	2
Xrl a, #num	XOR each bit of acc with same bit of immediate num, stores result in acc	Immediate	2	1
Xrl a, add	XOR each bit of acc with same bit of content in add, stores result in acc	Direct	2	1
Xrl a, Rx	XOR each bit of acc with same bit of content of Rx, stores result in acc	Register	1	1
Xrl a, @Rp	XOR each bit of acc with same bit of content of add given by Rp, stores result in acc	Indirect	1	1
Xrl add, a	XOR each bit of acc with same bit of direct add num, stores result in add	Direct	2	1
Xrl add, #num	XOR each bit of direct add with same bit of immediate num, stores result in add	direct	3	2
Clr a	Clear each bit of acc	Direct	1	1
Cpl a	Complement each bit of acc	direct	1	1
Anl c, b	AND carry with given bit b, stores result in carry	--	2	2
Anl c, /b	AND carry with complement of given bit b, stores result in carry	--	2	2
Orl c, b	OR carry with given bit b, stores result in carry	--	2	2
Orl c, /b	OR carry with complement of given bit b, stores result in carry	--	2	2
Cpl c	Complement carry flag	--	1	1
Cpl b	Complement bit b	--	2	1

Clr c	Clear carry flag	--	1	1
Clr b	Clear given bit b	--	2	1
Mov c, b	Copy bit b to carry	--	2	1
Mov b, c	Copy carry to bit b	--	2	2
Setb c	Set carry flag	--	1	1
Setb b	Set bit b	--	2	1
Rl a	Rotate acc one bit left	--	1	1
Rr a	Rotate acc one bit right	--	1	1
Rlc a	Rotate acc one bit left with carry	--	1	1
Rrc a	Rotate acc one bit right with carry	--	1	1
Swap a	Exchange upper and lower nibble of acc	--	1	1

Arithmetic Instructions: -

Mnemonics	Operational description	Addressing mode	No. of bytes occupied	No. of cycles used
Inc a	Add 1 to acc	Register	1	1
Inc Rr	Add 1 to register Rr	Register	1	1
Inc add	Add 1 to the content of add	Direct	2	1
Inc @rp	Add 1 to the content of the address in Rp	indirect	1	1
Inc DPTR	Add 1 to DPTR	Register	1	2
dec a	Subtract 1 from acc	Register	1	1
dec Rr	Subtract 1 from Rr	Register	1	1
dec add	Subtract 1 from content of add	Direct	2	1
dec @rp	Subtract 1 from the content of address	indirect	1	1
Add a, #num	Add the immediate num with acc and stores result in acc	immediate	2	1
Add a, Rx	Add the data in Rx with acc and stores result in acc	Register	1	1
Add a, add	Add the data in add with acc and stores result in acc	Direct	2	1
Add a, @Rp	Add the data at the address in Rp with acc and stores result in acc	Indirect	1	1
Addc a,#num	Add the immediate num with acc and carry, stores result in acc	immediate	2	1

Addc a, Rx	Add the data in Rx with acc and carry, stores result in acc	Register	1	1
Addc a, add	Add the data in add with acc and carry, stores result in acc	Direct	2	1
Addc a, @Rp	Add the data at the address in Rp with acc and carry, stores result in acc	Indirect	1	1
Subb a, #num	Subtract immediate num and carry from acc; stores the result in acc	immediate	2	1
Subb a, add	Subtract the content of add and carry from acc; stores the result in acc	Register	1	1
Subb a, Rx	Subtract the data in Rx and carry from acc; stores the result in acc	Direct	2	1
Subb a, @Rp	Subtract the data at the address in Rp and carry from acc; stores the result in acc	Indirect	1	1
Mul ab	Multiply acc and register B. store the lower byte of result in acc and higher byte in B	---	1	4
div ab	divide acc by register B. store quotient in acc and remainder in B	---	1	4
Da a	After addition of two packed BCD numbers, adjust the sum to decimal format	---	1	1

Branching Instructions: -

Mnemonic	Operational description	No of bytes occupied	No. of cycles used
Jc label	Jump to label if carry is set to 1	2	2
Jnc label	Jump to label if carry is cleared to 0	2	2
Jb b,label	Jump to label if given bit is set to 1	3	2
Jnb b,label	Jump to label if given bit is cleared to 0	3	2
Jbc b,label	Jump to label if given bit is set. Clear the bit	3	2

Cjne a, add, label	Compare the content of accumulator with the content of given address and if not equal jump to label	3	2
Cjne a, #num, label	Compare the content of accumulator with immediate number and if not equal jump to label	3	2
Cjne Rx, #num, label	Compare the content of Rx with the immediate number and if not equal jump to label	3	2
Cjne @Rp, #num, label	Compare the content of location in Rp with immediate number and if not equal jump to label	3	2
Djnz Rx, label	Decrement the content of Rx and jump to the label if it is not zero	2	2
Djnz add, label	Decrement the content of address and jump to the label if it is not zero	3	2
Jz label	Jump to the label if content of accumulator is 0	2	2
Jnz label	Jump to the label if content of accumulator is not 0	2	2
Jmp @a+dptr	Jump to the address created by adding the contents on accumulator and dptr	1	2
Ajmp sadd	Take a jump to absolute short range address sadd	2	2
Ljmp ladd	Take a jump to absolute long range address sadd	3	2
Sjmp radd	Take a jump to relative address radd	2	2
nop	Short form of no operation means do nothing and go to next instruction	1	1
Acall sadd	Pushes the content of Acc on stack and load it will absolute short range address sadd	2	2
Lcall ladd	Pushes the content of Acc on stack and load it will absolute long range address sadd	3	2
Ret	returns from subroutine by restoring the Acc from stack using pop operation	1	2
reti	Returns from interrupt subroutine by restoring Acc from stack using pop operation	1	2

Appendix B: ARM Instruction Set

The ARM instruction set summary is given below.

Operation		Assembly syntax
Move	Move	MOV{cond}{S} Rd, <0prnd2>
	Move NOT	MVN{cond}{S} Rd, <0prnd2>
	Move SPSR to register	MRS{cond} Rd, SPSR
	Move CPSR to register	MRS{cond} Rd, CPSR
	Move register to SPSR	MSR{cond} SPSR{field}, Rm
	Move register to CPSR	MSR{cond} CPSR{field}, Rm
	Move immediate to SPSR flags	MSR{cond} SPSR_f, #32bit_Imm
	Move immediate to CPSR flags	MSR{cond} CPSR_f, #32bit_Imm
Arithmetic	Add	ADD{cond}{S} Rd, Rn, <0prnd2>
	Add with carry	ADC{cond}{S} Rd, Rn, <0prnd2>
	Subtract	SUB{cond}{S} Rd, Rn, <0prnd2>
	Subtract with carry	SBC{cond}{S} Rd, Rn, <0prnd2>
	Subtract reverse subtract	RSB{cond}{S} Rd, Rn, <0prnd2>
	Subtract reverse subtract with carry	RSC{cond}{S} Rd, Rn, <0prnd2>
	Multiply	MUL{cond}{S} Rd, Rm, Rs
	Multiply accumulate	MLA{cond}{S} Rd, Rm, Rs, Rn
	Multiply unsigned long	UMULL{cond}{S} RdLo, RdHi, Rm, Rs
	Multiply unsigned accumulate long	UMLAL{cond}{S} RdLo, RdHi, Rm, Rs
	Multiply signed long	SMULL{cond}{S} RdLo, RdHi, Rm, Rs
	Multiply signed accumulate long	SMLAL{cond}{S} RdLo, RdHi, Rm, Rs
	Compare	CMP{cond} Rd, <0prnd2>
	Compare negative	CMN{cond} Rd, <0prnd2>
Logical	Test	TST{cond} Rn, <0prnd2>

Operation		Assembly syntax
	Test equivalence	TEQ{cond} Rn, <0prnd2>
	AND	AND{cond}{S} Rd, Rn, <0prnd2>
	EOR	EOR{cond}{S} Rd, Rn, <0prnd2>
	ORR	ORR{cond}{S} Rd, Rn, <0prnd2>
	Bit clear	BIC{cond}{S} Rd, Rn, <0prnd2>
Branch	Branch	B{cond} label
	Branch with link	BL{cond} label
	Branch and exchange instruction set	BX{cond} Rn
Load	Word	LDR{cond} Rd, <a_mode2>
	Word with user-mode privilege	LDR{cond}T Rd, <a_mode2P>
	Byte	LDR{cond}B Rd, <a_mode2>
	Byte with user-mode privilege	LDR{cond}BT Rd, <a_mode2P>
	Byte signed	LDR{cond}SB Rd, <a_mode3>
	Halfword	LDR{cond}H Rd, <a_mode3>
	Halfword signed	LDR{cond}SH Rd, <a_mode3>
	Multiple block data operations	-
	• Increment before	LDM{cond}IB Rd{!}, <reglist>{^}
	• Increment after	LDM{cond}IA Rd{!}, <reglist>{^}
	• Decrement before	LDM{cond}DB Rd{!}, <reglist>{^}
	• Decrement after	LDM{cond}DA Rd{!}, <reglist>{^}
	• Stack operation	LDM{cond}<a_mode4L> Rd{!}, <reglist>
	• Stack operation, and restore CPSR	LDM{cond}<a_mode4L> Rd{!}, <reglist+pc>^
	• Stack operation with user registers	LDM{cond}<a_mode4L> Rd{!}, <reglist>^
Store	Word	STR{cond} Rd, <a_mode2>
	Word with user-mode privilege	STR{cond}T Rd, <a_mode2P>

Operation		Assembly syntax
	Byte	STR{cond}B Rd, <a_mode2>
	Byte with user-mode privilege	STR{cond}BT Rd, <a_mode2P>
	Halfword	STR{cond}H Rd, <a_mode3>
	Multiple block data operations	-
	• Increment before	STM{cond}IB Rd{!}, <reglist>{^}
	• Increment after	STM{cond}IA Rd{!}, <reglist>{^}
	• Decrement before	STM{cond}DB Rd{!}, <reglist>{^}
	• Decrement after	STM{cond}DA Rd{!}, <reglist>{^}
	• Stack operation	STM{cond}<a_mode4S> Rd{!}, <reglist>
	• Stack operation with user registers	STM{cond}<a_mode4S> Rd{!}, <reglist>^
Swap	Word	SWP{cond} Rd, Rm, [Rn]
	Byte	SWP{cond}B Rd, Rm, [Rn]
Coprocessors	Data operation	CDP{cond} p<cpnum>, <op1>, CRd, CRn, CRm, <op2>
	Move to ARM register from coprocessor	MRC{cond} p<cpnum>, <op1>, Rd, CRn, CRm, <op2>
	Move to coprocessor from ARM register	MCR{cond} p<cpnum>, <op1>, Rd, CRn, CRm, <op2>
	Load	LDC{cond} p<cpnum>, CRd, <a_mode5>
	Store	STC{cond} p<cpnum>, CRd, <a_mode5>
Software interrupt		SWI 24bit_Imm

Addressing modes

The addressing modes are procedures shared by different instructions for generating values used by the instructions. The five addressing modes used by the ARM7TDMI processor are:

Mode 1 Shifter operands for data processing instructions.

Mode 2 Load and store word or unsigned byte.

Mode 3 Load and store halfword or load signed byte.

Mode 4 Load and store multiple.

Mode 5 Load and store coprocessor.

Appendix C: ASCII Codes

This table lists the ASCII characters and their decimal, octal and hexadecimal numbers. Characters which appear as names in parentheses (*e.g.*, (nl)) are non-printing characters. A table of the common non-printing characters appears after this table.

Char	Dec	Oct	Hex	Char	Dec	Oct	Hex	Char	Dec	Oct	Hex	Char	Dec	Oct	He
(nul)	0	0000	0x00	(sp)	32	0040	0x20	@	64	0100	0x40	`	96	0140	0x6
(soh)	1	0001	0x01	!	33	0041	0x21	A	65	0101	0x41	a	97	0141	0x6
(stx)	2	0002	0x02	"	34	0042	0x22	B	66	0102	0x42	b	98	0142	0x6
(etx)	3	0003	0x03	#	35	0043	0x23	C	67	0103	0x43	c	99	0143	0x6
(eot)	4	0004	0x04	$	36	0044	0x24	D	68	0104	0x44	d	100	0144	0x6
(enq)	5	0005	0x05	%	37	0045	0x25	E	69	0105	0x45	e	101	0145	0x6
(ack)	6	0006	0x06	&	38	0046	0x26	F	70	0106	0x46	f	102	0146	0x6
(bel)	7	0007	0x07	'	39	0047	0x27	G	71	0107	0x47	g	103	0147	0x6
(bs)	8	0010	0x08	(	40	0050	0x28	H	72	0110	0x48	h	104	0150	0x6
(ht)	9	0011	0x09	)	41	0051	0x29	I	73	0111	0x49	i	105	0151	0x6
(nl)	10	0012	0x0a	*	42	0052	0x2a	J	74	0112	0x4a	j	106	0152	0x6
(vt)	11	0013	0x0b	+	43	0053	0x2b	K	75	0113	0x4b	k	107	0153	0x6
(np)	12	0014	0x0c	,	44	0054	0x2c	L	76	0114	0x4c	l	108	0154	0x6
(cr)	13	0015	0x0d	-	45	0055	0x2d	M	77	0115	0x4d	m	109	0155	0x6
(so)	14	0016	0x0e	.	46	0056	0x2e	N	78	0116	0x4e	n	110	0156	0x6
(si)	15	0017	0x0f	/	47	0057	0x2f	O	79	0117	0x4f	o	111	0157	0x6
(dle)	16	0020	0x10	0	48	0060	0x30	P	80	0120	0x50	p	112	0160	0x7
(dc1)	17	0021	0x11	1	49	0061	0x31	Q	81	0121	0x51	q	113	0161	0x7
(dc2)	18	0022	0x12	2	50	0062	0x32	R	82	0122	0x52	r	114	0162	0x7
(dc3)	19	0023	0x13	3	51	0063	0x33	S	83	0123	0x53	s	115	0163	0x7
(dc4)	20	0024	0x14	4	52	0064	0x34	T	84	0124	0x54	t	116	0164	0x7
(nak)	21	0025	0x15	5	53	0065	0x35	U	85	0125	0x55	u	117	0165	0x7
(syn)	22	0026	0x16	6	54	0066	0x36	V	86	0126	0x56	v	118	0166	0x7
(etb)	23	0027	0x17	7	55	0067	0x37	W	87	0127	0x57	w	119	0167	0x7
(can)	24	0030	0x18	8	56	0070	0x38	X	88	0130	0x58	x	120	0170	0x7
(em)	25	0031	0x19	9	57	0071	0x39	Y	89	0131	0x59	y	121	0171	0x7
(sub)	26	0032	0x1a	:	58	0072	0x3a	Z	90	0132	0x5a	z	122	0172	0x7
(esc)	27	0033	0x1b	;	59	0073	0x3b	[	91	0133	0x5b	{	123	0173	0x7
(fs)	28	0034	0x1c	<	60	0074	0x3c	\	92	0134	0x5c	|	124	0174	0x7
(gs)	29	0035	0x1d	=	61	0075	0x3d	]	93	0135	0x5d	}	125	0175	0x7
(rs)	30	0036	0x1e	>	62	0076	0x3e	^	94	0136	0x5e	~	126	0176	0x7
(us)	31	0037	0x1f	?	63	0077	0x3f	_	95	0137	0x5f	(del)	127	0177	0x7

ASCII Name	Description	C Escape Sequence
nul	null byte	\0
bel	bell character	\a
bs	backspace	\b
ht	horizontal tab	\t
np	formfeed	\f
nl	newline	\n
cr	carriage return	\r
vt	vertical tab	
esc	escape	
sp	space	

Appendix D: Data Sheets

The datasheets of the components used in the presented work is provided in the Compact Disc along with the thesis. Below is the list of datasheets:

1. 1N4007.pdf (Diode)
2. 1N4148.pdf (Diode)
3. 74LS244.pdf (Buffer IC)
4. 89c2051.pdf (Microcontroller IC)
5. AT89C51.pdf (Microcontroller IC)
6. AT89C52.pdf (Microcontroller IC)
7. CD4017BC.pdf (Decade Counter IC)
8. LCD.pdf (LCD Module)
9. LCD_16x1.pdf (LCD Module)
10. hef4519b.pdf (2-input Multiplexer)
11. HT12D.pdf (Decoder IC)
12. ht12e.pdf (Encoder IC)
13. LPC2119-2129 Chip Details.pdf (ARM processor)
14. MAX3232.pdf (RS-232 Line Driver)
15. MCP255.pdf (High Speed CAN Transceiver)
16. Philips_P89C51RD2_6.pdf (Microcontroller IC)

Appendix E: References

1. The 8051 microcontroller & embedded systems By *M.A.Mazidi and J.G.Mazidi*
2. The microcontroller idea book By *Jan Axelson*
3. Assembly language programming By *L S electronic systems.*
4. Embedded microcontrollers hand book By *Intel*
5. 8051 cross assembler user's manual: *MetaLink Corporation Chandler, Arizona.*
6. *Philips Semiconductors*: http://www.semiconductors.philips.com
7. P89C51RD2BN/01 - 80C51 8-bit Flash microcontroller family Data sheet - Philips Semiconductors.
8. AN461 In-circuit and In-application programming of the 9C51Rx+/Rx2/66x microcontrollers, Philips Semiconductors, filename: AN461_7.pdf
9. Download the latest WinISP version from http://www.semiconductors.philipsmcu.com/flash/flash.html.
10. Phyton Inc. Microsystems and Development Tools, www.phyton.com.
11. Applications of Radio Frequency Identification (RFID), Anthony Sabetti, Texas Instruments Registration and Identification Systems, SCAN-TECH 94.
12. Radio Frequency Identification Basics for Manufacturing, G. Dan Sutton, President Balogh Corporation, SCAN-TECH 93.
13. Embedded C By Michael J. Pont
14. Programming and Customizing the 8051 Microcontroller By Myke Predko.
15. ARM System-on-Chip Architecture By Steve Furber.
16. ARM System Developer's Guide: Designing and Optimizing System Software By Andrew N. Sloss, Dominic Symes, Chris Wright.
17. Let Us C by Yashavant Kanetkar.
18. Real Time Systems By Rajib Mall
19. The 8051 Microcontroller Architecture, Programming & Applications By Kenneth J. Ayala.

FAIRCHILD
SEMICONDUCTOR®

1N4001 - 1N4007

Features

- Low forward voltage drop.
- High surge current capability.

DO-41
COLOR BAND DENOTES CATHODE

General Purpose Rectifiers (Glass Passivated)

Absolute Maximum Ratings*

T_A = 25°C unless otherwise noted

Symbol	Parameter	Value							Units
		4001	4002	4003	4004	4005	4006	4007	
V_{RRM}	Peak Repetitive Reverse Voltage	50	100	200	400	600	800	1000	V
$I_{F(AV)}$	Average Rectified Forward Current, .375 " lead length @ T_A = 75°C	1.0							A
I_{FSM}	Non-repetitive Peak Forward Surge Current 8.3 ms Single Half-Sine-Wave	30							A
T_{stg}	Storage Temperature Range	-55 to +175							°C
T_J	Operating Junction Temperature	-55 to +175							°C

*These ratings are limiting values above which the serviceability of any semiconductor device may be impaired.

Thermal Characteristics

Symbol	Parameter	Value	Units
P_D	Power Dissipation	3.0	W
$R_{\theta JA}$	Thermal Resistance, Junction to Ambient	50	°C/W

Electrical Characteristics

T_A = 25°C unless otherwise noted

Symbol	Parameter	Device							Units
		4001	4002	4003	4004	4005	4006	4007	
V_F	Forward Voltage @ 1.0 A	1.1							V
I_{rr}	Maximum Full Load Reverse Current, Full Cycle T_A = 75°C	30							µA
I_R	Reverse Current @ rated V_R T_A = 25°C T_A = 100°C	5.0 500							µA µA
C_T	Total Capacitance V_R = 4.0 V, f = 1.0 MHz	15							pF

Typical Characteristics

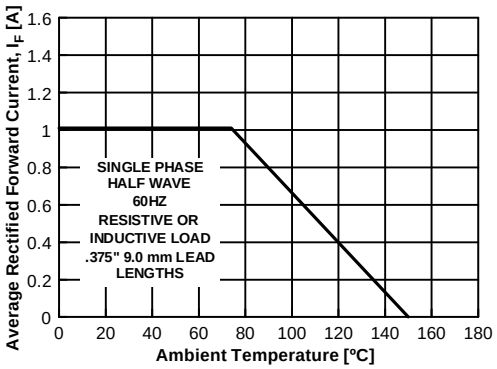


Figure 1. Forward Current Derating Curve

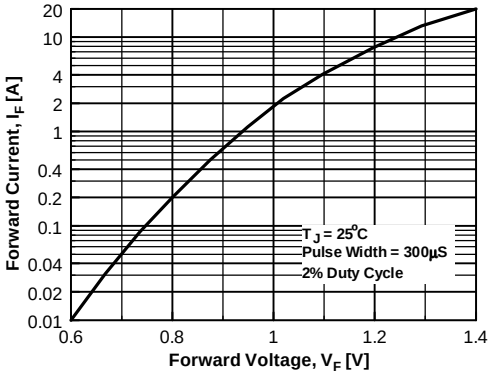


Figure 2. Forward Voltage Characteristics

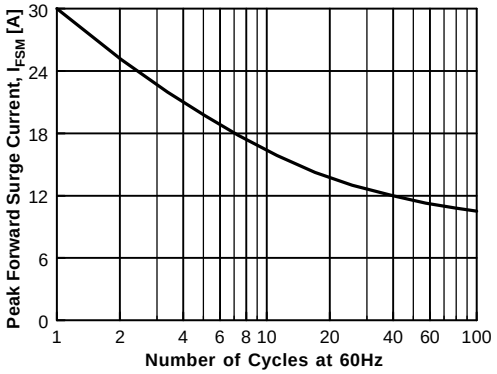


Figure 3. Non-Repetitive Surge Current

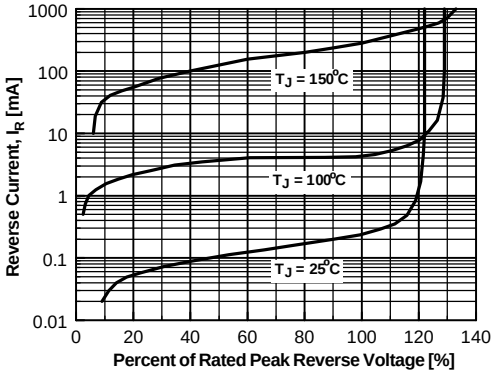


Figure 4. Reverse Current vs Reverse Voltage

TRADEMARKS

The following are registered and unregistered trademarks Fairchild Semiconductor owns or is authorized to use and is not intended to be an exhaustive list of all such trademarks.

ACEx™	FAST ®	OPTOLOGIC™	SMART START™	VCX™
Bottomless™	FASTr™	OPTOPLANAR™	STAR*POWER™	
CoolFET™	FRFET™	PACMAN™	Stealth™	
CROSSVOLT™	GlobalOptoisolator™	POP™	SuperSOT™-3	
DenseTrench™	GTO™	Power247™	SuperSOT™-6	
DOME™	HiSeC™	PowerTrench®	SuperSOT™-8	
EcoSPARK™	ISOPLANAR™	QFET™	SyncFET™	
E^2CMOS™	LittleFET™	QS™	TinyLogic™	
EnSigna™	MicroFET™	QT Optoelectronics™	TruTranslation™	
FACT™	MicroPak™	Quiet Series™	UHC™	
FACT Quiet Series™	MICROWIRE™	SILENT SWITCHER®	UltraFET®	

STAR*POWER is used under license

DISCLAIMER

FAIRCHILD SEMICONDUCTOR RESERVES THE RIGHT TO MAKE CHANGES WITHOUT FURTHER NOTICE TO ANY PRODUCTS HEREIN TO IMPROVE RELIABILITY, FUNCTION OR DESIGN. FAIRCHILD DOES NOT ASSUME ANY LIABILITY ARISING OUT OF THE APPLICATION OR USE OF ANY PRODUCT OR CIRCUIT DESCRIBED HEREIN; NEITHER DOES IT CONVEY ANY LICENSE UNDER ITS PATENT RIGHTS, NOR THE RIGHTS OF OTHERS.

LIFE SUPPORT POLICY

FAIRCHILD'S PRODUCTS ARE NOT AUTHORIZED FOR USE AS CRITICAL COMPONENTS IN LIFE SUPPORT DEVICES OR SYSTEMS WITHOUT THE EXPRESS WRITTEN APPROVAL OF FAIRCHILD SEMICONDUCTOR CORPORATION.
As used herein:

1. Life support devices or systems are devices or systems which, (a) are intended for surgical implant into the body, or (b) support or sustain life, or (c) whose failure to perform when properly used in accordance with instructions for use provided in the labeling, can be reasonably expected to result in significant injury to the user.

2. A critical component is any component of a life support device or system whose failure to perform can be reasonably expected to cause the failure of the life support device or system, or to affect its safety or effectiveness.

PRODUCT STATUS DEFINITIONS

Definition of Terms

Datasheet Identification	Product Status	Definition
Advance Information	Formative or In Design	This datasheet contains the design specifications for product development. Specifications may change in any manner without notice.
Preliminary	First Production	This datasheet contains preliminary data, and supplementary data will be published at a later date. Fairchild Semiconductor reserves the right to make changes at any time without notice in order to improve design.
No Identification Needed	Full Production	This datasheet contains final specifications. Fairchild Semiconductor reserves the right to make changes at any time without notice in order to improve design.
Obsolete	Not In Production	This datasheet contains specifications on a product that has been discontinued by Fairchild semiconductor. The datasheet is printed for reference information only.

FAIRCHILD
SEMICONDUCTOR®

1N/FDLL 914/A/B / 916/A/B / 4148 / 4448

DO-35

LL-34

THE PLACEMENT OF THE EXPANSION GAP HAS NO RELATIONSHIP TO THE LOCATION OF THE CATHODE TERMINAL

COLOR BAND MARKING

DEVICE	1ST BAND	2ND BAND
FDLL914	BLACK	BROWN
FDLL914A	BLACK	GRAY
FDLL914B	BROWN	BLACK
FDLL916	BLACK	RED
FDLL916A	BLACK	WHITE
FDLL916B	BROWN	BROWN
FDLL4148	BLACK	BROWN
FDLL4448	BROWN	BLACK

Small Signal Diode

Absolute Maximum Ratings*

T_A = 25°C unless otherwise noted

Symbol	Parameter	Value	Units
V_{RRM}	Maximum Repetitive Reverse Voltage	100	V
$I_{F(AV)}$	Average Rectified Forward Current	200	mA
I_{FSM}	Non-repetitive Peak Forward Surge Current Pulse Width = 1.0 second Pulse Width = 1.0 microsecond	 1.0 4.0	 A A
T_{stg}	Storage Temperature Range	-65 to +200	°C
T_J	Operating Junction Temperature	175	°C

*These ratings are limiting values above which the serviceability of any semiconductor device may be impaired.

NOTES:
1) These ratings are based on a maximum junction temperature of 200 degrees C.
2) These are steady state limits. The factory should be consulted on applications involving pulsed or low duty cycle operations.

Thermal Characteristics

Symbol	Characteristic	Max	Units
		1N/FDLL 914/A/B / 4148 / 4448	
P_D	Power Dissipation	500	mW
$R_{\theta JA}$	Thermal Resistance, Junction to Ambient	300	°C/W

Small Signal Diode
(continued)

Electrical Characteristics
T_A = 25°C unless otherwise noted

Symbol	Parameter		Test Conditions	Min	Max	Units
V_R	Breakdown Voltage		I_R = 100 µA	100		V
			I_R = 5.0 µA	75		V
V_F	Forward Voltage	**1N914B/4448**	I_F = 5.0 mA	620	720	mV
		1N916B	I_F = 5.0 mA	630	730	mV
		1N914/916/4148	I_F = 10 mA		1.0	V
		1N914A/916A	I_F = 20 mA		1.0	V
		1N916B	I_F = 20 mA		1.0	V
		1N914B/4448	I_F = 100 mA		1.0	V
I_R	Reverse Current		V_R = 20 V		25	nA
			V_R = 20 V, T_A = 150°C		50	µA
			V_R = 75 V		5.0	µA
C_T	Total Capacitance					
		1N916A/B/4448	V_R = 0, f = 1.0 MHz		2.0	pF
		1N914A/B/4148	V_R = 0, f = 1.0 MHz		4.0	pF
t_{rr}	Reverse Recovery Time		I_F = 10 mA, V_R = 6.0 V (60mA), I_{rr} = 1.0 mA, R_L = 100Ω		4.0	ns

Typical Characteristics

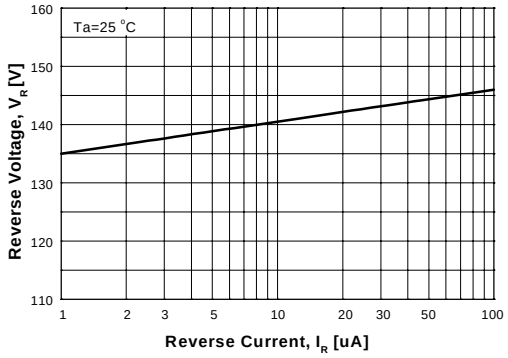
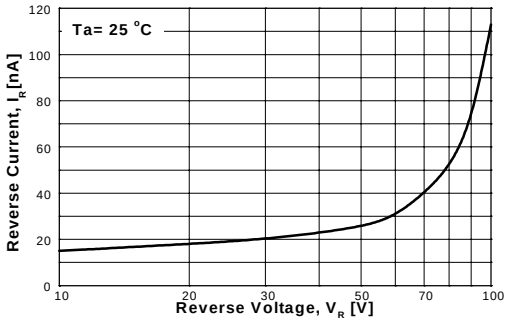


Figure 1. Reverse Voltage vs Reverse Current
BV - 1.0 to 100 uA



GENERAL RULE: The Reverse Current of a diode will approximately double for every ten (10) Degree C increase in Temperature

Figure 2. Reverse Current vs Reverse Voltage
IR - 10 to 100 V

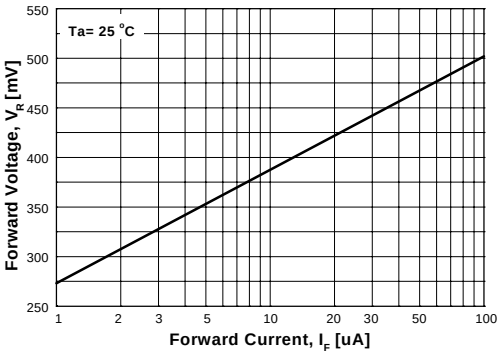


Figure 3. Forward Voltage vs Forward Current
VF - 1 to 100 uA

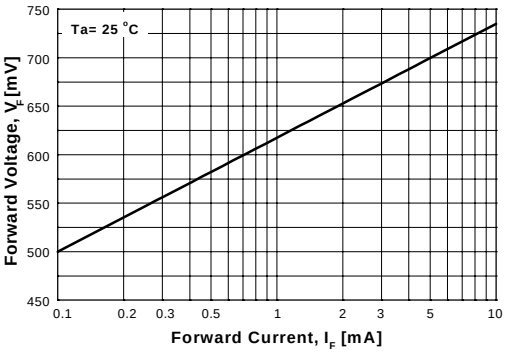


Figure 4. Forward Voltage vs Forward Current
VF - 0.1 to 10 mA

Typical Characteristics (continued)

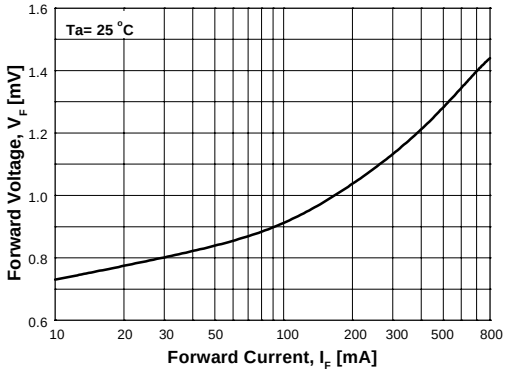


Figure 5. Forward Voltage vs Forward Current VF - 10 to 800 mA

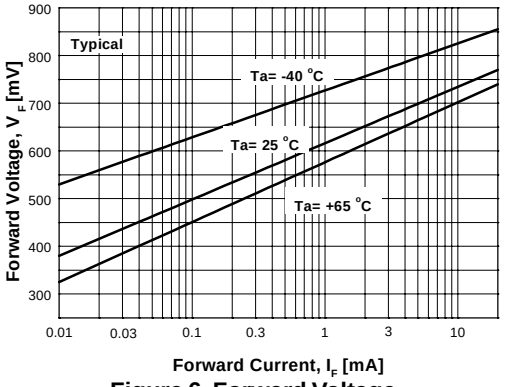


Figure 6. Forward Voltage vs Ambient Temperature VF - 0.01 - 20 mA (-40 to +65 Deg C)

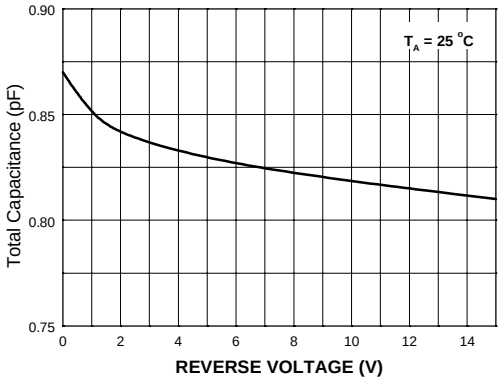


Figure 7. Total Capacitance

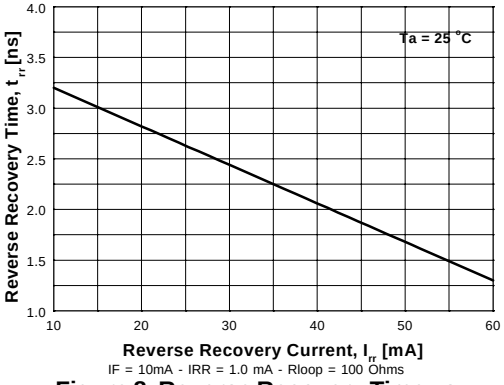


Figure 8. Reverse Recovery Time vs Reverse Recovery Current

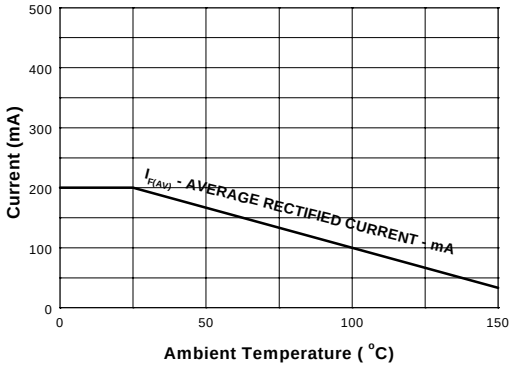


Figure 9. Average Rectified Current ($I_{F(AV)}$) versus Ambient Temperature (T_A)

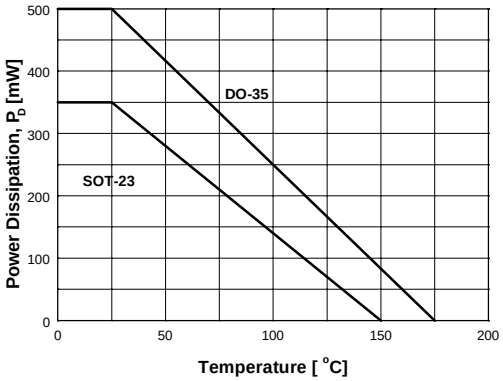


Figure 10. Power Derating Curve

TRADEMARKS

The following are registered and unregistered trademarks Fairchild Semiconductor owns or is authorized to use and is not intended to be an exhaustive list of all such trademarks.

ACEx™	FAST ®	MICROWIRE™	SILENT SWITCHER ®	UHC™
Bottomless™	FASTr™	OPTOLOGIC ®	SMART START™	UltraFET ®
CoolFET™	FRFET™	OPTOPLANAR™	SPM™	VCX™
CROSSVOLT™	GlobalOptoisolator™	PACMAN™	STAR*POWER™	
DenseTrench™	GTO™	POP™	Stealth™	
DOME™	HiSeC™	Power247™	SuperSOT™-3	
EcoSPARK™	I^2C™	PowerTrench ®	SuperSOT™-6	
E^2CMOS^{TM}	ISOPLANAR™	QFET™	SuperSOT™-8	
$EnSigna^{TM}$	LittleFET™	QS™	SyncFET™	
FACT™	MicroFET™	QT Optoelectronics™	TinyLogic™	
FACT Quiet Series™	MicroPak™	Quiet Series™	TruTranslation™	

STAR*POWER is used under license

DISCLAIMER

FAIRCHILD SEMICONDUCTOR RESERVES THE RIGHT TO MAKE CHANGES WITHOUT FURTHER NOTICE TO ANY PRODUCTS HEREIN TO IMPROVE RELIABILITY, FUNCTION OR DESIGN. FAIRCHILD DOES NOT ASSUME ANY LIABILITY ARISING OUT OF THE APPLICATION OR USE OF ANY PRODUCT OR CIRCUIT DESCRIBED HEREIN; NEITHER DOES IT CONVEY ANY LICENSE UNDER ITS PATENT RIGHTS, NOR THE RIGHTS OF OTHERS.

LIFE SUPPORT POLICY

FAIRCHILD'S PRODUCTS ARE NOT AUTHORIZED FOR USE AS CRITICAL COMPONENTS IN LIFE SUPPORT DEVICES OR SYSTEMS WITHOUT THE EXPRESS WRITTEN APPROVAL OF FAIRCHILD SEMICONDUCTOR CORPORATION.
As used herein:

1. Life support devices or systems are devices or systems which, (a) are intended for surgical implant into the body, or (b) support or sustain life, or (c) whose failure to perform when properly used in accordance with instructions for use provided in the labeling, can be reasonably expected to result in significant injury to the user.

2. A critical component is any component of a life support device or system whose failure to perform can be reasonably expected to cause the failure of the life support device or system, or to affect its safety or effectiveness.

PRODUCT STATUS DEFINITIONS

Definition of Terms

Datasheet Identification	Product Status	Definition
Advance Information	Formative or In Design	This datasheet contains the design specifications for product development. Specifications may change in any manner without notice.
Preliminary	First Production	This datasheet contains preliminary data, and supplementary data will be published at a later date. Fairchild Semiconductor reserves the right to make changes at any time without notice in order to improve design.
No Identification Needed	Full Production	This datasheet contains final specifications. Fairchild Semiconductor reserves the right to make changes at any time without notice in order to improve design.
Obsolete	Not In Production	This datasheet contains specifications on a product that has been discontinued by Fairchild semiconductor. The datasheet is printed for reference information only.

M74LS244P

OCTAL BUFFER/LINE DRIVERS WITH 3-STATE OUTPUTS(NONINVERTED)

DESCRIPTION

The M74LS244P is a semiconductor integrated circuit containing 2 blocks of buffers with 3-state non-inverted output and common output controlling input for all 4 discrete circuits.

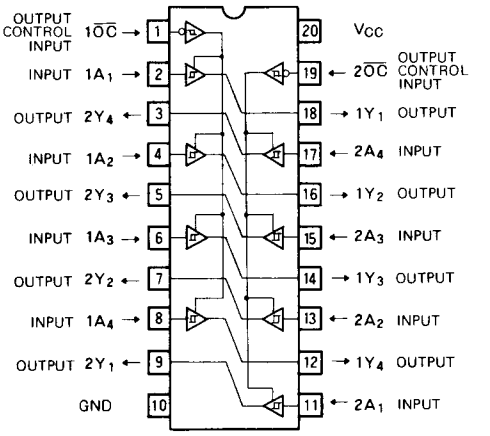
FEATURES

- Low input load factor (pnp input)
- Hysteresis provided (= 400mV typical)
- High breakdown input voltage ($V_I \geqq 15V$)
- Output control input having same phase for 2 circuits
- High fan-out, 3-state output
  (I_{OL} = 24mA, I_{OH} = -15mA)
- Wide operating temperature range (T_a = -20 ~ +75°C)

APPLICATION

General purpose, for use in industrial and consumer equipment.

PIN CONFIGURATION (TOP VIEW)



Outline 20P4

FUNCTIONAL DESCRIPTION

The use of pnp transistors in the input circuit has enabled the achievement of small input load factor. With hysteresis characteristics, the buffer has a 3-state noninverted output with high noise margin.

When output control input $\overline{OC}$ is low, the output Y is low if input A is low and Y is high if A is high. When $\overline{OC}$ is high, all of Y_1, Y_2, Y_3, and Y_4 are in the high-impedance state, irrespective of the status of A.

By connecting $1\overline{OC}$ with $2\overline{OC}$, it becomes possible to control the output of all 8 circuits simultaneously. Output can be terminated by a load resistor of 133Ω or over.

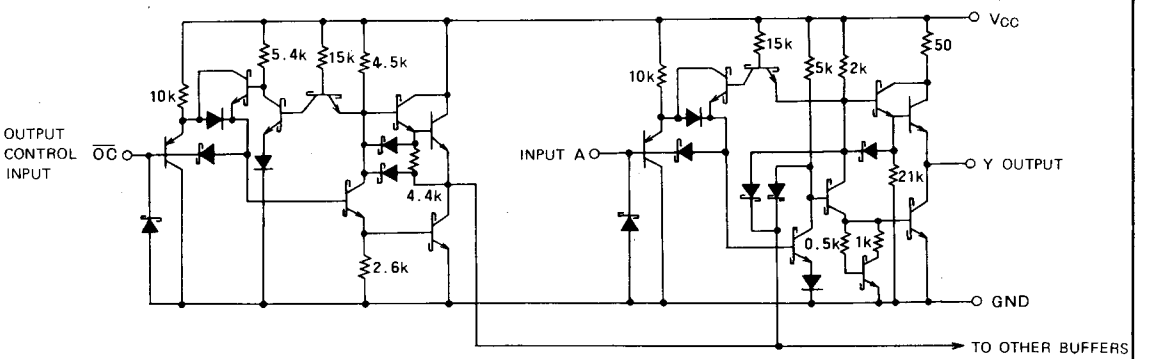
For standard characteristics, see M74LS241P.

FUNCTION TABLE (Note 1)

A	$\overline{OC}$	Y
L	L	L
H	L	H
X	H	Z

Note 1: Z : high-impedance
X : irrelevant

CIRCUIT DIAGRAM (EACH BUFFER)



UNIT: Ω

OCTAL BUFFER/LINE DRIVERS WITH 3-STATE OUTPUTS(NONINVERTED)

ABSOLUTE MAXIMUM RATINGS (Ta = −20 ~ +75°C, unless otherwise noted)

Symbol	Parameter	Conditions	Limits	Unit
V_{CC}	Supply voltage		−0.5 ~ +7	V
V_I	Input voltage		−0.5 ~ +15	V
V_O	Output voltage	Off-state	−0.5 ~ +5.5	V
Topr	Operating free-air ambient temperature range		−20 ~ +75	°C
Tstg	Storage temperature range		−65 ~ +150	°C

RECOMMENDED OPERATING CONDITIONS (Ta = −20 ~ +75°C, unless otherwise noted)

Symbol	Parameter		Limits			Unit
			Min	Typ	Max	
V_{CC}	Supply voltage		4.75	5	5.25	V
I_{OH}	High-level output current	$V_{OH} \geqq 2.4V$			−3	mA
		$V_{OH} \geqq 2V$			−15	mA
I_{OL}	Low-level output current	$V_{OL} \leqq 0.4V$			12	mA
		$V_{OL} \leqq 0.5V$			24	mA

ELECTRICAL CHARACTERISTICS (Ta = −20 ~ +75°C, unless otherwise noted)

Symbol	Parameter	Test conditions		Limits			Unit
				Min	Typ*	Max	
V_{IH}	High-level input voltage			2			V
V_{IL}	Low-level input voltage					0.8	V
$V_{T+} - V_{T-}$	Hysteresis	V_{CC}=4.75V		0.2	0.4		V
V_{IC}	Input clamp voltage	V_{CC}=4.75V, I_{IC}=−18mA				−1.5	V
V_{OH}	High-level output voltage	V_{CC}=4.75V, V_I=2V	V_I=0.8V, I_{OH}=−3mA	2.4	3.4		V
			V_I=0.5V, I_{OH}=−15mA	2			V
V_{OL}	Low-level output voltage	V_{CC}=4.75V, V_I=0.8V, V_I=2V	I_{OL}=12mA		0.25	0.4	V
			I_{OL}=24mA		0.35	0.5	V
I_{OZH}	Off-state high-level output current	V_{CC}=5.25V, V_I=2V, V_O=2.7V				20	μA
I_{OZL}	Off-state low-level output current	V_{CC}=5.25V, V_I=2V, V_O=0.4V				−20	μA
I_{IH}	High-level input current	V_{CC}=5.25V, V_I=2.7V				20	μA
		V_{CC}=5.25V, V_I=10V				0.1	mA
I_{IL}	Low-level input current	V_{CC}=5.25V, V_I=0.4V				−0.2	mA
I_{OS}	Short-circuit output current (Note 2)	V_{CC}=5.25V, V_O=0V		−40		−225	mA
I_{CCH}	Supply current, all outputs high	V_{CC}=5.25V, V_I=0V, V_I=4.5V			17	27	mA
I_{CCL}	Supply current, all outputs low	V_{CC}=5.25V, V_I=0V			27	46	mA
I_{CCZ}	Supply current, all outputs off	V_{CC}=5.25V, V_I=4.5V			32	54	mA

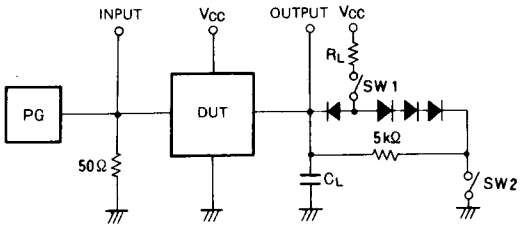
* : All typical values are at V_{CC} = 5V, Ta = 25°C.
Note 2: All measurements should be done quickly, and not more than one output should be shorted at a time.

SWITCHING CHARACTERISTICS (V_{CC}=5V, Ta = 25°C, unless otherwise noted)

Symbol	Parameter	Test conditions	Limits			Unit
			Min	Typ	Max	
t_{PLH}	Low-to-high-level, high-to-low-level output propagation time, from input A to output Y	C_L=45pF (Note 3)		8	18	ns
t_{PHL}				9	18	ns
t_{PZH}	Output enable time to high-level	R_L=667Ω, C_L=45pF (Note 3)		15	30	ns
t_{PZL}	Output enable time to low-level	R_L=667Ω, C_L=45pF (Note 3)		12	40	ns
t_{PLZ}	Output disable time from low-level	R_L=667Ω, C_L= 5 pF (Note 3)		11	25	ns
t_{PHZ}	Output disable time from high-level	R_L=667Ω, C_L= 5 pF (Note 3)		12	18	ns

OCTAL BUFFER/LINE DRIVERS WITH 3-STATE OUTPUTS(NONINVERTED)

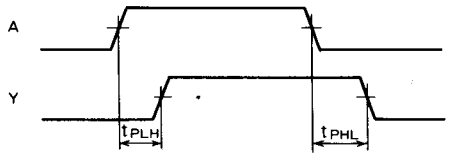
Note 3: Measurement circuit

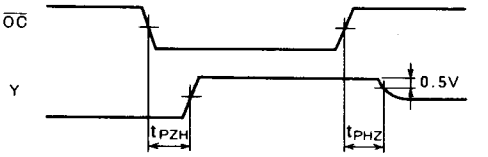


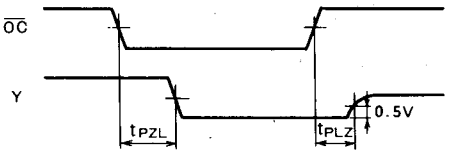
Symbol	SW 1	SW2
t_{PZH}	Open	Closed
t_{PZL}	Closed	Open
t_{PLZ}	Closed	Closed
t_{PHZ}	Closed	Closed

(1) The pulse generator (PG) has the following characteristics: PRR = 1MHz, t_r = 6ns, t_f = 6ns, t_W = 500ns, V_P = 3V_{P-P}, Z_O = 50Ω
(2) All diodes are switching diodes ($t_{rr} \leqq$ 4ns)
(3) C_L includes probe and jig capacitance.

TIMING DIAGRAM (Reference level = 1.3V)

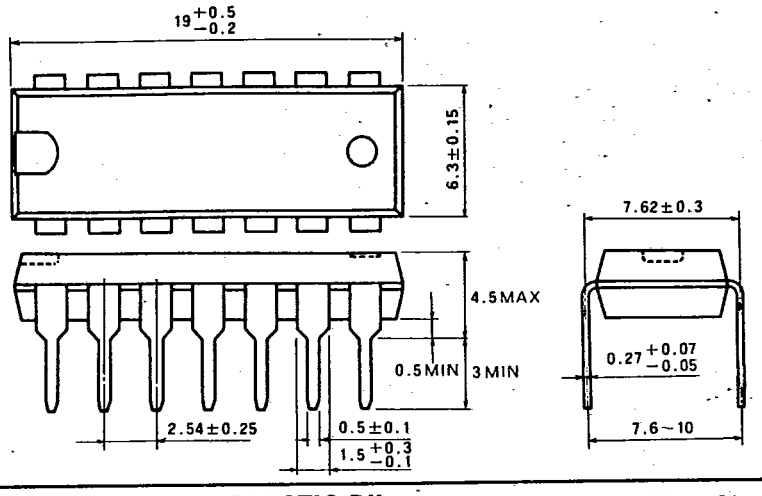






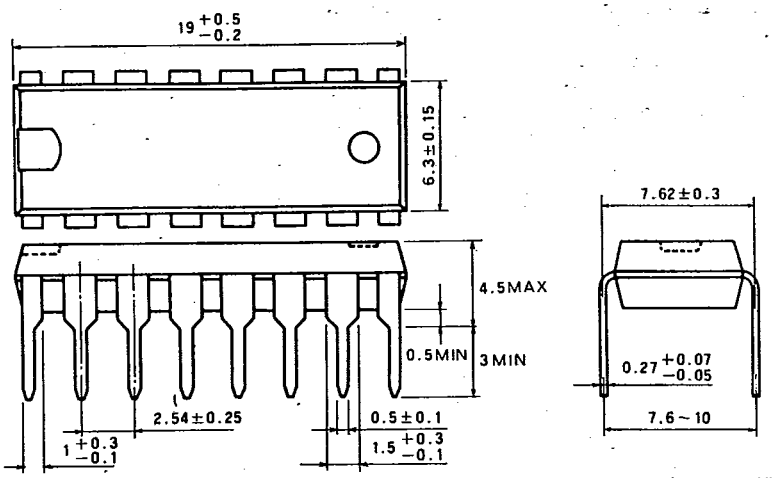
TYPE 14P4 14-PIN MOLDED PLASTIC DIL

Dimension in mm



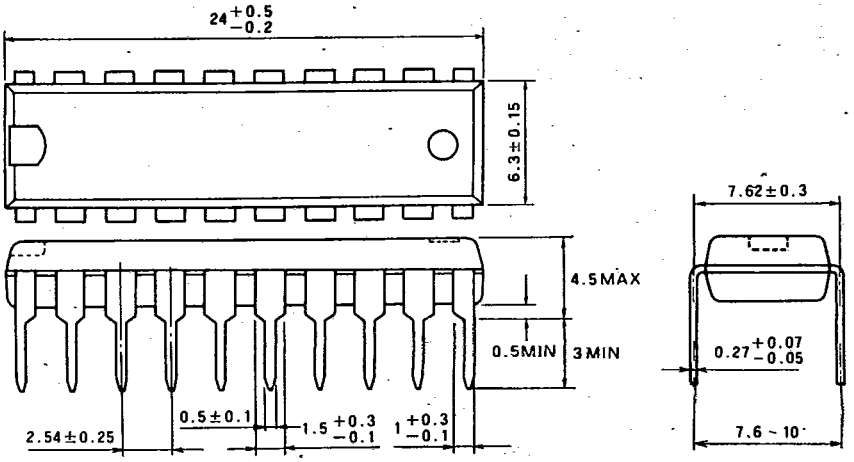
TYPE 16P4 16-PIN MOLDED PLASTIC DIL

Dimension in mm



TYPE 20P4 20-PIN MOLDED PLASTIC DIL

Dimension in mm



Features

- **Compatible with MCS-51™ Products**
- **2K Bytes of Reprogrammable Flash Memory**
  - **Endurance: 1,000 Write/Erase Cycles**
- **2.7V to 6V Operating Range**
- **Fully Static Operation: 0 Hz to 24 MHz**
- **Two-Level Program Memory Lock**
- **128 x 8-Bit Internal RAM**
- **15 Programmable I/O Lines**
- **Two 16-Bit Timer/Counters**
- **Six Interrupt Sources**
- **Programmable Serial UART Channel**
- **Direct LED Drive Outputs**
- **On-Chip Analog Comparator**
- **Low Power Idle and Power Down Modes**

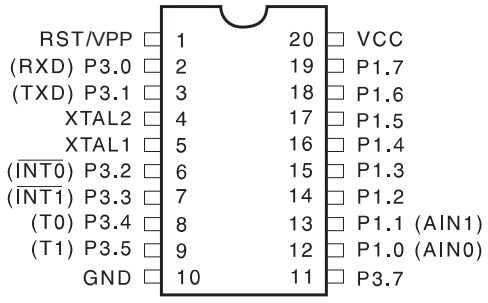
Description

The AT89C2051 is a low-voltage, high-performance CMOS 8-bit microcomputer with 2K Bytes of Flash programmable and erasable read only memory (PEROM). The device is manufactured using Atmel's high density nonvolatile memory technology and is compatible with the industry standard MCS-51™ instruction set. By combining a versatile 8-bit CPU with Flash on a monolithic chip, the Atmel AT89C2051 is a powerful microcomputer which provides a highly flexible and cost effective solution to many embedded control applications.

The AT89C2051 provides the following standard features: 2K Bytes of Flash, 128 bytes of RAM, 15 I/O lines, two 16-bit timer/counters, a five vector two-level interrupt architecture, a full duplex serial port, a precision analog comparator, on-chip oscillator and clock circuitry. In addition, the AT89C2051 is designed with static logic for operation down to zero frequency and supports two software selectable power saving modes. The Idle Mode stops the CPU while allowing the RAM, timer/counters, serial port and interrupt system to continue functioning. The Power Down Mode saves the RAM contents but freezes the oscillator disabling all other chip functions until the next hardware reset.

Pin Configuration

PDIP/SOIC

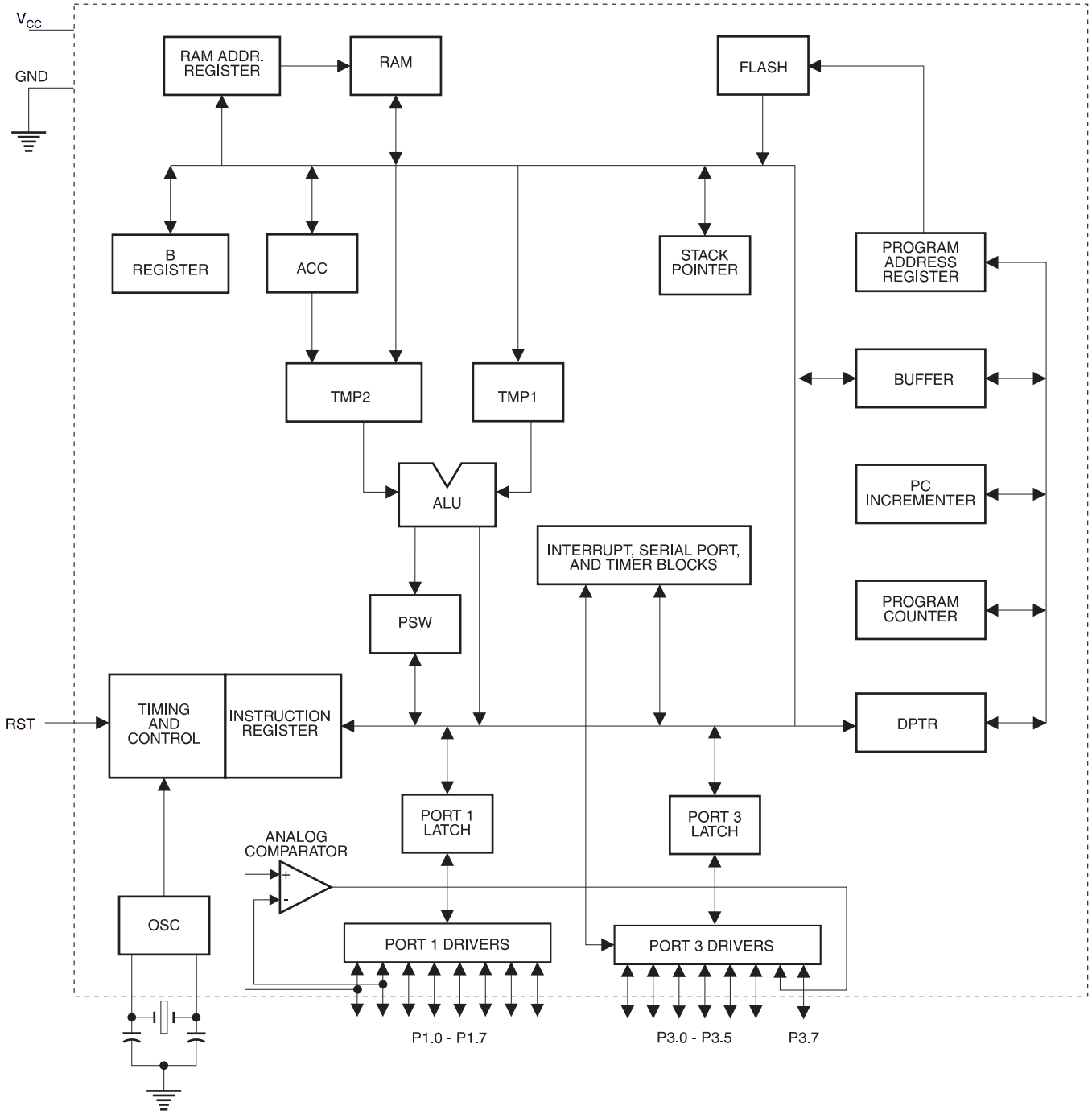


8-Bit Microcontroller with 2K Bytes Flash

AT89C2051

0368D-B–12/97

Block Diagram



Pin Description

V_{CC}
Supply voltage.

GND
Ground.

Port 1
Port 1 is an 8-bit bidirectional I/O port. Port pins P1.2 to P1.7 provide internal pullups. P1.0 and P1.1 require external pullups. P1.0 and P1.1 also serve as the positive input (AIN0) and the negative input (AIN1), respectively, of the on-chip precision analog comparator. The Port 1 output buffers can sink 20 mA and can drive LED displays directly. When 1s are written to Port 1 pins, they can be used as inputs. When pins P1.2 to P1.7 are used as inputs and are externally pulled low, they will source current (I_{IL}) because of the internal pullups.

Port 1 also receives code data during Flash programming and verification.

Port 3
Port 3 pins P3.0 to P3.5, P3.7 are seven bidirectional I/O pins with internal pullups. P3.6 is hard-wired as an input to the output of the on-chip comparator and is not accessible as a general purpose I/O pin. The Port 3 output buffers can sink 20 mA. When 1s are written to Port 3 pins they are pulled high by the internal pullups and can be used as inputs. As inputs, Port 3 pins that are externally being pulled low will source current (I_{IL}) because of the pullups.

Port 3 also serves the functions of various special features of the AT89C2051 as listed below:

Port Pin	Alternate Functions
P3.0	RXD (serial input port)
P3.1	TXD (serial output port)
P3.2	$\overline{INT0}$ (external interrupt 0)
P3.3	$\overline{INT1}$ (external interrupt 1)
P3.4	T0 (timer 0 external input)
P3.5	T1 (timer 1 external input)

Port 3 also receives some control signals for Flash programming and verification.

RST
Reset input. All I/O pins are reset to 1s as soon as RST goes high. Holding the RST pin high for two machine cycles while the oscillator is running resets the device.

Each machine cycle takes 12 oscillator or clock cycles.

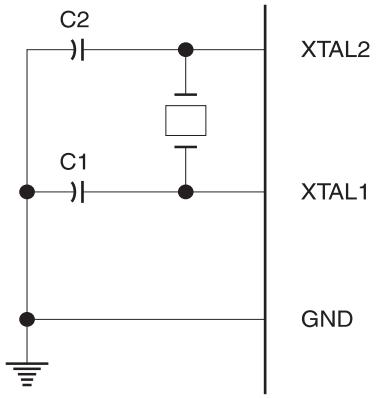
XTAL1
Input to the inverting oscillator amplifier and input to the internal clock operating circuit.

XTAL2
Output from the inverting oscillator amplifier.

Oscillator Characteristics

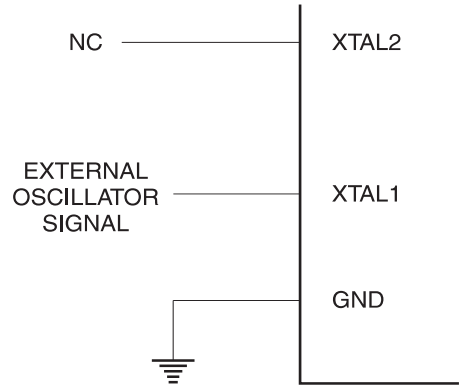
XTAL1 and XTAL2 are the input and output, respectively, of an inverting amplifier which can be configured for use as an on-chip oscillator, as shown in Figure 1. Either a quartz crystal or ceramic resonator may be used. To drive the device from an external clock source, XTAL2 should be left unconnected while XTAL1 is driven as shown in Figure 2. There are no requirements on the duty cycle of the external clock signal, since the input to the internal clocking circuitry is through a divide-by-two flip-flop, but minimum and maximum voltage high and low time specifications must be observed.

Figure 1. Oscillator Connections



Note: C1, C2 = 30 pF ± 10 pF for Crystals
= 40 pF ± 10 pF for Ceramic Resonators

Figure 2. External Clock Drive Configuration



Special Function Registers

A map of the on-chip memory area called the Special Function Register (SFR) space is shown in the table below.

Note that not all of the addresses are occupied, and unoccupied addresses may not be implemented on the chip. Read accesses to these addresses will in general return random data, and write accesses will have an indeterminate effect.

User software should not write 1s to these unlisted locations, since they may be used in future products to invoke new features. In that case, the reset or inactive values of the new bits will always be 0.

Table 1. AT89C2051 SFR Map and Reset Values

0F8H									0FFH
0F0H	B 00000000								0F7H
0E8H									0EFH
0E0H	ACC 00000000								0E7H
0D8H									0DFH
0D0H	PSW 00000000								0D7H
0C8H									0CFH
0C0H									0C7H
0B8H	IP XXX00000								0BFH
0B0H	P3 11111111								0B7H
0A8H	IE 0XX00000								0AFH
0A0H									0A7H
98H	SCON 00000000	SBUF XXXXXXXX							9FH
90H	P1 11111111								97H
88H	TCON 00000000	TMOD 00000000	TL0 00000000	TL1 00000000	TH0 00000000	TH1 00000000			8FH
80H		SP 00000111	DPL 00000000	DPH 00000000				PCON 0XXX0000	87H

Restrictions on Certain Instructions

The AT89C2051 and is an economical and cost-effective member of Atmel's growing family of microcontrollers. It contains 2K bytes of flash program memory. It is fully compatible with the MCS-51 architecture, and can be programmed using the MCS-51 instruction set. However, there are a few considerations one must keep in mind when utilizing certain instructions to program this device.

All the instructions related to jumping or branching should be restricted such that the destination address falls within the physical program memory space of the device, which is 2K for the AT89C2051. This should be the responsibility of the software programmer. For example, LJMP 7E0H would be a valid instruction for the AT89C2051 (with 2K of memory), whereas LJMP 900H would not.

1. Branching instructions:
LCALL, LJMP, ACALL, AJMP, SJMP, JMP @A+DPTR

These unconditional branching instructions will execute correctly as long as the programmer keeps in mind that the destination branching address must fall within the physical boundaries of the program memory size (locations 00H to 7FFH for the 89C2051). Violating the physical space limits may cause unknown program behavior.

CJNE [...], DJNZ [...], JB, JNB, JC, JNC, JBC, JZ, JNZ With these conditional branching instructions the same rule above applies. Again, violating the memory boundaries may cause erratic execution.

For applications involving interrupts the normal interrupt service routine address locations of the 80C51 family architecture have been preserved.

2. MOVX-related instructions, Data Memory:
The AT89C2051 contains 128 bytes of internal data memory. Thus, in the AT89C2051 the stack depth is limited to 128 bytes, the amount of available RAM. External DATA memory access is not supported in this device, nor is external PROGRAM memory execution. Therefore, no MOVX [...] instructions should be included in the program.

A typical 80C51 assembler will still assemble instructions, even if they are written in violation of the restrictions mentioned above. It is the responsibility of the controller user to know the physical features and limitations of the device being used and adjust the instructions used correspondingly.

Program Memory Lock Bits

On the chip are two lock bits which can be left unprogrammed (U) or can be programmed (P) to obtain the additional features listed in the table below:

Lock Bit Protection Modes(1)

Program Lock Bits			
	LB1	LB2	Protection Type
1	U	U	No program lock features.
2	P	U	Further programming of the Flash is disabled.
3	P	P	Same as mode 2, also verify is disabled.

Note: 1. The Lock Bits can only be erased with the Chip Erase operation.

Idle Mode

In idle mode, the CPU puts itself to sleep while all the on-chip peripherals remain active. The mode is invoked by software. The content of the on-chip RAM and all the special functions registers remain unchanged during this mode. The idle mode can be terminated by any enabled interrupt or by a hardware reset.

P1.0 and P1.1 should be set to '0' if no external pullups are used, or set to '1' if external pullups are used.

It should be noted that when idle is terminated by a hardware reset, the device normally resumes program execution, from where it left off, up to two machine cycles before the internal reset algorithm takes control. On-chip hardware inhibits access to internal RAM in this event, but access to the port pins is not inhibited. To eliminate the possibility of an unexpected write to a port pin when Idle is terminated by reset, the instruction following the one that invokes Idle should not be one that writes to a port pin or to external memory.

Power Down Mode

In the power down mode the oscillator is stopped, and the instruction that invokes power down is the last instruction executed. The on-chip RAM and Special Function Registers retain their values until the power down mode is terminated. The only exit from power down is a hardware reset. Reset redefines the SFRs but does not change the on-chip RAM. The reset should not be activated before V_{CC} is restored to its normal operating level and must be held active long enough to allow the oscillator to restart and stabilize.

P1.0 and P1.1 should be set to '0' if no external pullups are used, or set to '1' if external pullups are used.

Programming The Flash

The AT89C2051 is shipped with the 2K bytes of on-chip PEROM code memory array in the erased state (i.e., contents = FFH) and ready to be programmed. The code memory array is programmed one byte at a time. *Once the array is programmed, to re-program any non-blank byte, the entire memory array needs to be erased electrically.*

Internal Address Counter: The AT89C2051 contains an internal PEROM address counter which is always reset to 000H on the rising edge of RST and is advanced by applying a positive going pulse to pin XTAL1.

Programming Algorithm: To program the AT89C2051, the following sequence is recommended.

1. Power-up sequence:
   Apply power between V_{CC} and GND pins
   Set RST and XTAL1 to GND
2. Set pin RST to 'H'
   Set pin P3.2 to 'H'
3. Apply the appropriate combination of 'H' or 'L' logic levels to pins P3.3, P3.4, P3.5, P3.7 to select one of the programming operations shown in the PEROM Programming Modes table.

To Program and Verify the Array:

4. Apply data for Code byte at location 000H to P1.0 to P1.7.
5. Raise RST to 12V to enable programming.
6. Pulse P3.2 once to program a byte in the PEROM array or the lock bits. The byte-write cycle is self-timed and typically takes 1.2 ms.
7. To verify the programmed data, lower RST from 12V to logic 'H' level and set pins P3.3 to P3.7 to the appropiate levels. Output data can be read at the port P1 pins.
8. To program a byte at the next address location, pulse XTAL1 pin once to advance the internal address counter. Apply new data to the port P1 pins.
9. Repeat steps 5 through 8, changing data and advancing the address counter for the entire 2K bytes array or until the end of the object file is reached.
10. Power-off sequence:
    set XTAL1 to 'L'
    set RST to 'L'
    Turn V_{CC} power off

$\overline{\text{Data}}$ Polling: The AT89C2051 features $\overline{\text{Data}}$ Polling to indicate the end of a write cycle. During a write cycle, an attempted read of the last byte written will result in the complement of the written data on P1.7. Once the write cycle has been completed, true data is valid on all outputs, and the next cycle may begin. $\overline{\text{Data}}$ Polling may begin any time after a write cycle has been initiated.

Ready/$\overline{\text{Busy}}$: The Progress of byte programming can also be monitored by the RDY/$\overline{\text{BSY}}$ output signal. Pin P3.1 is pulled low after P3.2 goes High during programming to indicate BUSY. P3.1 is pulled High again when programming is done to indicate READY.

Program Verify: If lock bits LB1 and LB2 have not been programmed code data can be read back via the data lines for verification:

1. Reset the internal address counter to 000H by bringing RST from 'L' to 'H'.
2. Apply the appropriate control signals for Read Code data and read the output data at the port P1 pins.
3. Pulse pin XTAL1 once to advance the internal address counter.
4. Read the next code data byte at the port P1 pins.
5. Repeat steps 3 and 4 until the entire array is read.

The lock bits cannot be verified directly. Verification of the lock bits is achieved by observing that their features are enabled.

Chip Erase: The entire PEROM array (2K bytes) and the two Lock Bits are erased electrically by using the proper combination of control signals and by holding P3.2 low for 10 ms. The code array is written with all "1"s in the Chip Erase operation and must be executed before any non-blank memory byte can be re-programmed.

Reading the Signature Bytes: The signature bytes are read by the same procedure as a normal verification of locations 000H, 001H, and 002H, except that P3.5 and P3.7 must be pulled to a logic low. The values returned are as follows.

(000H) = 1EH indicates manufactured by Atmel
(001H) = 21H indicates 89C2051

Programming Interface

Every code byte in the Flash array can be written and the entire array can be erased by using the appropriate combination of control signals. The write operation cycle is self-timed and once initiated, will automatically time itself to completion.

All major programming vendors offer worldwide support for the Atmel microcontroller series. Please contact your local programming vendor for the appropriate software revision.

Flash Programming Modes

Mode		RST/VPP	P3.2/$\overline{PROG}$	P3.3	P3.4	P3.5	P3.7
Write Code Data(1)(3)		12V		L	H	H	H
Read Code Data(1)		H	H	L	L	H	H
Write Lock	Bit - 1	12V		H	H	H	H
	Bit - 2	12V		H	H	L	L
Chip Erase		12V	(2)	H	L	L	L
Read Signature Byte		H	H	L	L	L	L

Notes:
1. The internal PEROM address counter is reset to 000H on the rising edge of RST and is advanced by a positive pulse at XTAL 1 pin.
2. Chip Erase requires a 10-ms $\overline{PROG}$ pulse.
3. P3.1 is pulled Low during programming to indicate RDY/$\overline{BSY}$.

Figure 3. Programming the Flash Memory

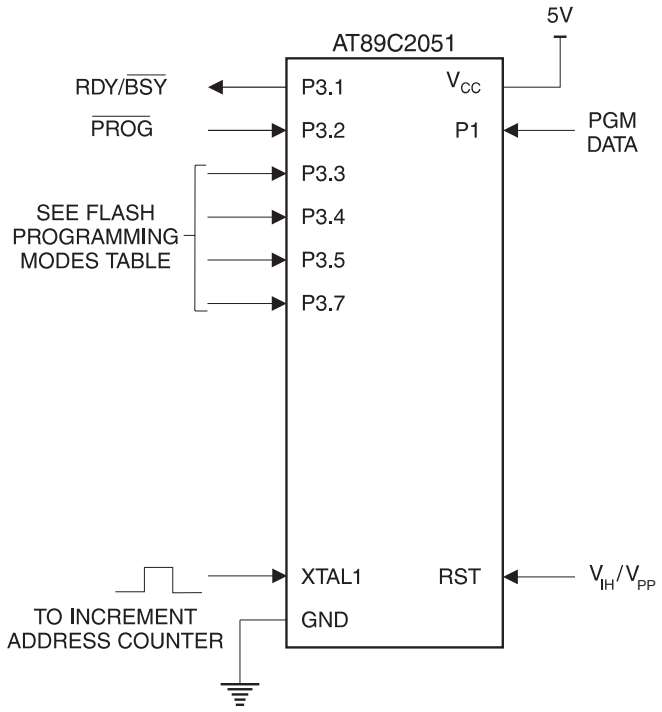
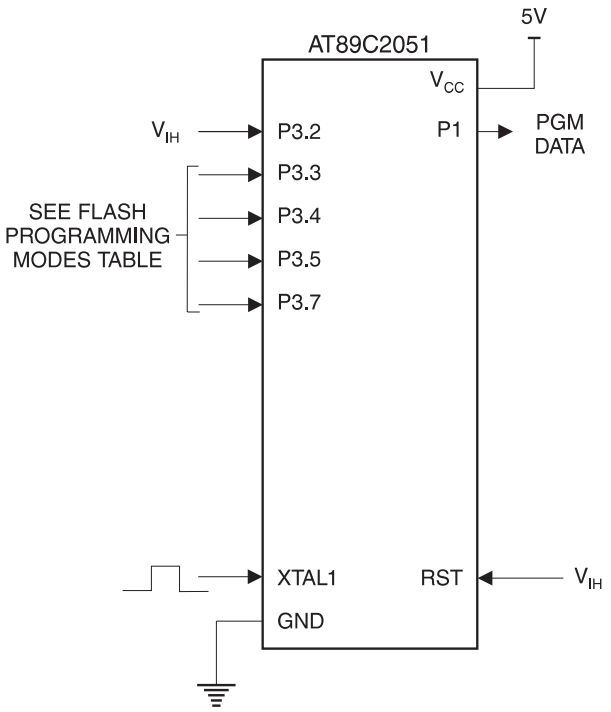


Figure 4. Verifying the Flash Memory



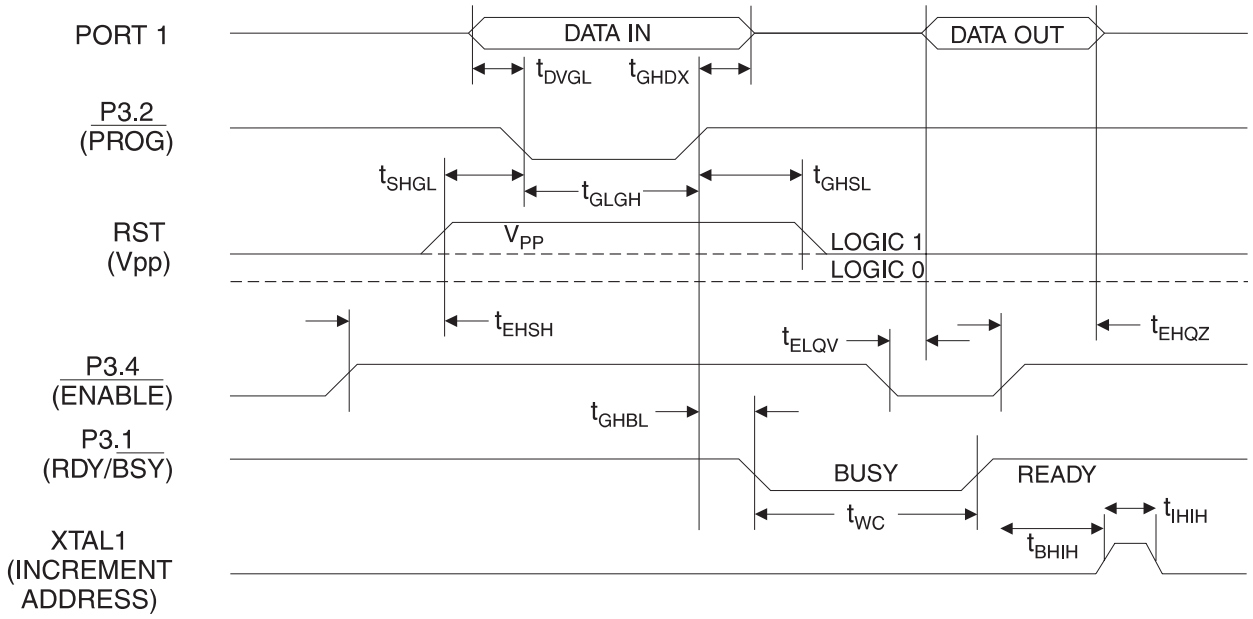
Flash Programming and Verification Characteristics

T_A = 0°C to 70°C, V_{CC} = 5.0 ± 10%

Symbol	Parameter	Min	Max	Units
V_{PP}	Programming Enable Voltage	11.5	12.5	V
I_{PP}	Programming Enable Current		250	µA
t_{DVGL}	Data Setup to $\overline{PROG}$ Low	1.0		µs
t_{GHDX}	Data Hold After $\overline{PROG}$	1.0		µs
t_{EHSH}	P3.4 ($\overline{ENABLE}$) High to V_{PP}	1.0		µs
t_{SHGL}	V_{PP} Setup to $\overline{PROG}$ Low	10		µs
t_{GHSL}	V_{PP} Hold After $\overline{PROG}$	10		µs
t_{GLGH}	$\overline{PROG}$ Width	1	110	µs
t_{ELQV}	$\overline{ENABLE}$ Low to Data Valid		1.0	µs
t_{EHQZ}	Data Float After $\overline{ENABLE}$	0	1.0	µs
t_{GHBL}	$\overline{PROG}$ High to $\overline{BUSY}$ Low		50	ns
t_{WC}	Byte Write Cycle Time		2.0	ms
t_{BHIH}	RDY/$\overline{BSY}$\ to Increment Clock Delay	1.0		µs
t_{IHIL}	Increment Clock High	200		ns

Note: 1. Only used in 12-volt programming mode.

Flash Programming and Verification Waveforms



Absolute Maximum Ratings*

Operating Temperature	-55°C to +125°C
Storage Temperature	-65°C to +150°C
Voltage on Any Pin with Respect to Ground	-1.0V to +7.0V
Maximum Operating Voltage	6.6V
DC Output Current	25.0 mA

*NOTICE: Stresses beyond those listed under "Absolute Maximum Ratings" may cause permanent damage to the device. This is a stress rating only and functional operation of the device at these or any other conditions beyond those indicated in the operational sections of this specification is not implied. Exposure to absolute maximum rating conditions for extended periods may affect device reliability.

DC Characteristics

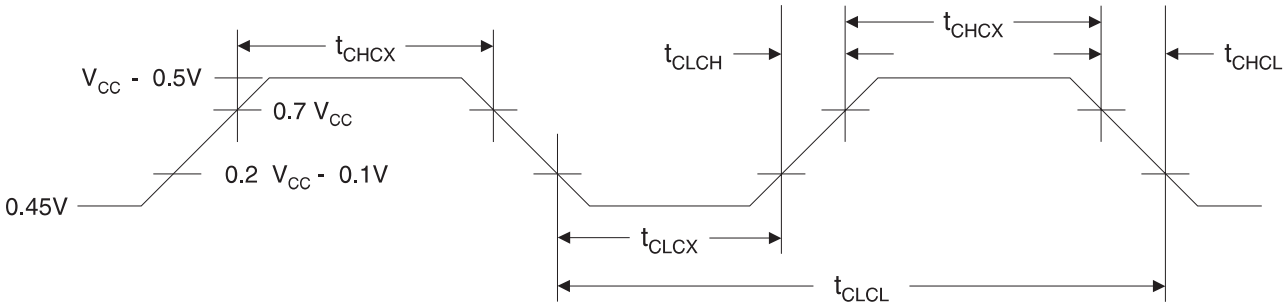
T_A = -40°C to 85°C, V_{CC} = 2.0V to 6.0V (unless otherwise noted)

Symbol	Parameter	Condition	Min	Max	Units
V_{IL}	Input Low Voltage		-0.5	0.2 V_{CC} - 0.1	V
V_{IH}	Input High Voltage	(Except XTAL1, RST)	0.2 V_{CC} + 0.9	V_{CC} + 0.5	V
V_{IH1}	Input High Voltage	(XTAL1, RST)	0.7 V_{CC}	V_{CC} + 0.5	V
V_{OL}	Output Low Voltage(1) (Ports 1, 3)	I_{OL} = 20 mA, V_{CC} = 5V I_{OL} = 10 mA, V_{CC} = 2.7V		0.5	V
V_{OH}	Output High Voltage (Ports 1, 3)	I_{OH} = -80 µA, V_{CC} = 5V ± 10%	2.4		V
		I_{OH} = -30 µA	0.75 V_{CC}		V
		I_{OH} = -12 µA	0.9 V_{CC}		V
I_{IL}	Logical 0 Input Current (Ports 1, 3)	V_{IN} = 0.45V		-50	µA
I_{TL}	Logical 1 to 0 Transition Current (Ports 1, 3)	V_{IN} = 2V, V_{CC} = 5V ± 10%		-750	µA
I_{LI}	Input Leakage Current (Port P1.0, P1.1)	0 < V_{IN} < V_{CC}		±10	µA
V_{OS}	Comparator Input Offset Voltage	V_{CC} = 5V		20	mV
V_{CM}	Comparator Input Common Mode Voltage		0	V_{CC}	V
RRST	Reset Pulldown Resistor		50	300	KΩ
C_{IO}	Pin Capacitance	Test Freq. = 1 MHz, T_A = 25°C		10	pF
I_{CC}	Power Supply Current	Active Mode, 12 MHz, V_{CC} = 6V/3V		15/5.5	mA
		Idle Mode, 12 MHz, V_{CC} = 6V/3V P1.0 & P1.1 = 0V or V_{CC}		5/1	mA
	Power Down Mode(2)	V_{CC} = 6V P1.0 & P1.1 = 0V or V_{CC}		100	µA
		V_{CC} = 3V P1.0 & P1.1 = 0V or V_{CC}		20	µA

Notes: 1. Under steady state (non-transient) conditions, I_{OL} must be externally limited as follows:
Maximum I_{OL} per port pin: 20 mA
Maximum total I_{OL} for all output pins: 80 mA
If I_{OL} exceeds the test condition, V_{OL} may exceed the related specification. Pins are not guaranteed to sink current greater than the listed test conditions.

2. Minimum V_{CC} for Power Down is 2V.

External Clock Drive Waveforms



External Clock Drive

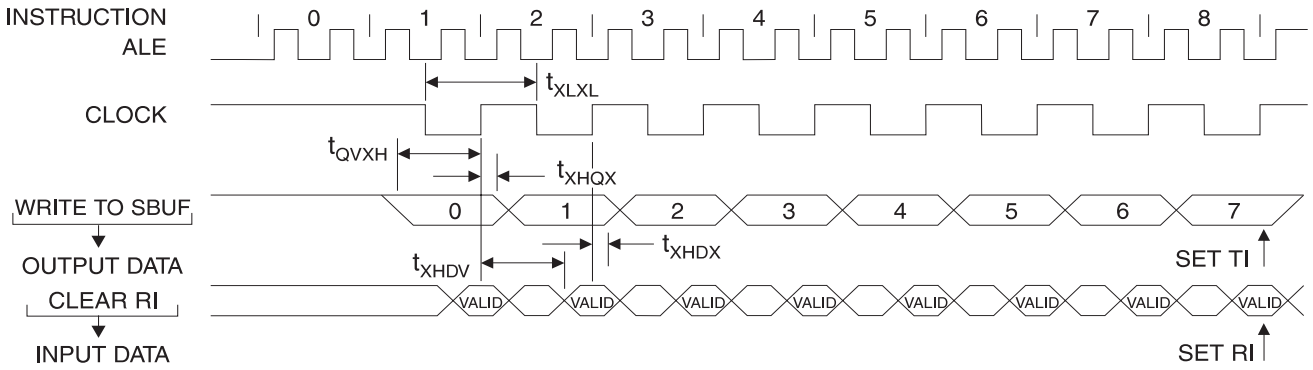
Symbol	Parameter	V_{CC} = 2.7V to 6.0V		V_{CC} = 4.0V to 6.0V		Units
		Min	Max	Min	Max	
$1/t_{CLCL}$	Oscillator Frequency	0	12	0	24	MHz
t_{CLCL}	Clock Period	83.3		41.6		ns
t_{CHCX}	High Time	30		15		ns
t_{CLCX}	Low Time	30		15		ns
t_{CLCH}	Rise Time		20		20	ns
t_{CHCL}	Fall Time		20		20	ns

Serial Port Timing: Shift Register Mode Test Conditions

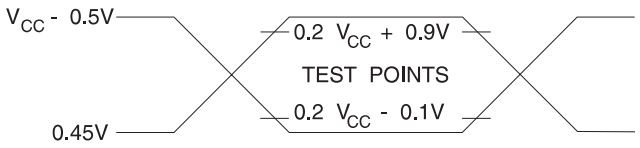
(V_{CC} = 5.0V ± 20%; Load Capacitance = 80 pF)

Symbol	Parameter	12 MHz Osc		Variable Oscillator		Units
		Min	Max	Min	Max	
t_{XLXL}	Serial Port Clock Cycle Time	1.0		$12t_{CLCL}$		µs
t_{QVXH}	Output Data Setup to Clock Rising Edge	700		$10t_{CLCL}$-133		ns
t_{XHQX}	Output Data Hold After Clock Rising Edge	50		$2t_{CLCL}$-117		ns
t_{XHDX}	Input Data Hold After Clock Rising Edge	0		0		ns
t_{XHDV}	Clock Rising Edge to Input Data Valid		700		$10t_{CLCL}$-133	ns

Shift Register Mode Timing Waveforms

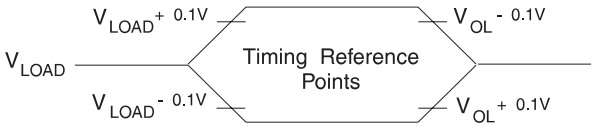


AC Testing Input/Output Waveforms(1)



Note: 1. AC Inputs during testing are driven at V_{CC} - 0.5V for a logic 1 and 0.45V for a logic 0. Timing measurements are made at V_{IH} min. for a logic 1 and V_{IL} max. for a logic 0.

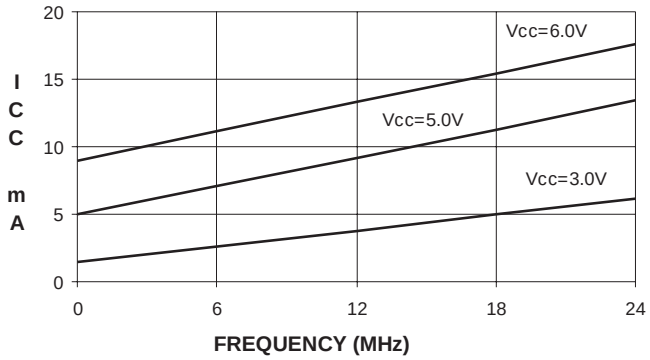
Float Waveforms(1)



Note: 1. For timing purposes, a port pin is no longer floating when a 100 mV change from load voltage occurs. A port pin begins to float when 100 mV change frothe loaded V_{OH}/V_{OL} level occurs.

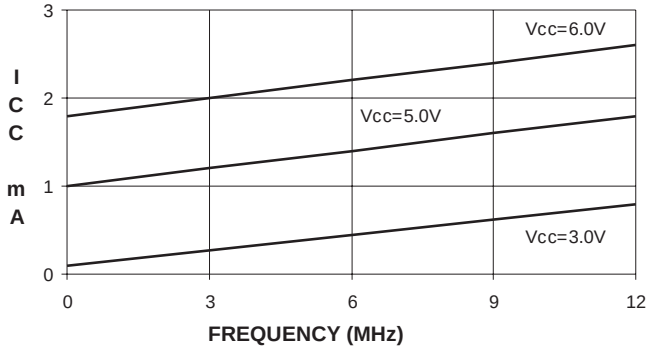
AT89C2051

TYPICAL ICC - ACTIVE (85°C)



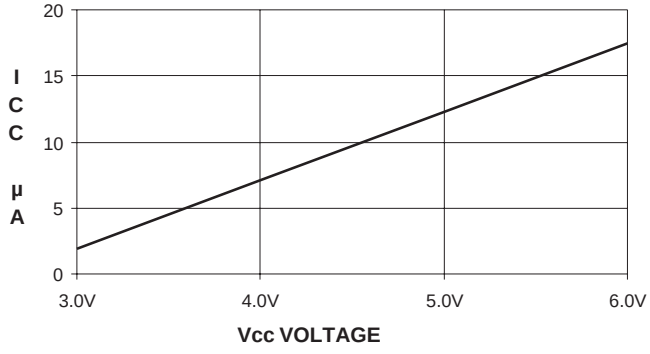
AT89C2051

TYPICAL ICC - IDLE (85°C)



AT89C2051

TYPICAL ICC vs. VOLTAGE- POWER DOWN (85°C)



Notes: 1. XTAL1 tied to GND for I_{CC} (power down)
2. P.1.0 and P1.1 = V_{CC} or GND
3. Lock bits programmed

Ordering Information

Speed (MHz)	Power Supply	Ordering Code	Package	Operation Range
12	2.7V to 6.0V	AT89C2051-12PC AT89C2051-12SC	20P3 20S	Commercial (0°C to 70°C)
		AT89C2051-12PI AT89C2051-12SI	20P3 20S	Industrial (-40°C to 85°C)
		AT89C2051-12PA AT89C2051-12SA	20P3 20S	Automotive (-40°C to 105°C)
24	4.0V to 6.0V	AT89C2051-24PC AT89C2051-24SC	20P3 20S	Commercial (0°C to 70°C)
		AT89C2051-24PI AT89C2051-24SI	20P3 20S	Industrial (-40°C to 85°C)

Package Type	
20P3	20 Lead, 0.300” Wide, Plastic Dual In-line Package (PDIP)
20S	20 Lead, 0.300” Wide, Plastic Gull Wing Small Outline (SOIC)

Features

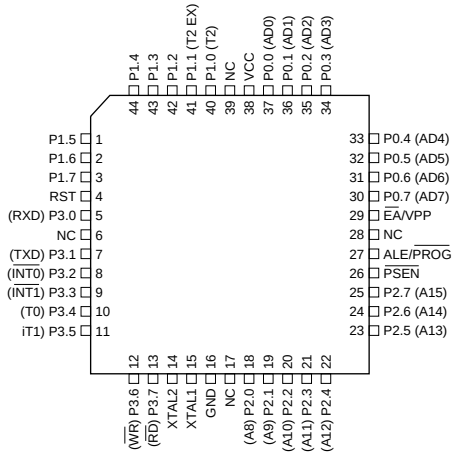
- **Compatible with MCS-51™ Products**
- **8K Bytes of In-System Reprogrammable Flash Memory**
- **Endurance: 1,000 Write/Erase Cycles**
- **Fully Static Operation: 0 Hz to 24 MHz**
- **Three-level Program Memory Lock**
- **256 x 8-bit Internal RAM**
- **32 Programmable I/O Lines**
- **Three 16-bit Timer/Counters**
- **Eight Interrupt Sources**
- **Programmable Serial Channel**
- **Low-power Idle and Power-down Modes**

Description

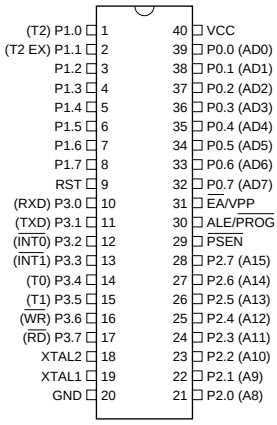
The AT89C52 is a low-power, high-performance CMOS 8-bit microcomputer with 8K bytes of Flash programmable and erasable read only memory (PEROM). The device is manufactured using Atmel's high-density nonvolatile memory technology and is compatible with the industry-standard 80C51 and 80C52 instruction set and pinout. The on-chip Flash allows the program memory to be reprogrammed in-system or by a conventional nonvolatile memory programmer. By combining a versatile 8-bit CPU with Flash on a monolithic chip, the Atmel AT89C52 is a powerful microcomputer which provides a highly-flexible and cost-effective solution to many embedded control applications.

Pin Configurations

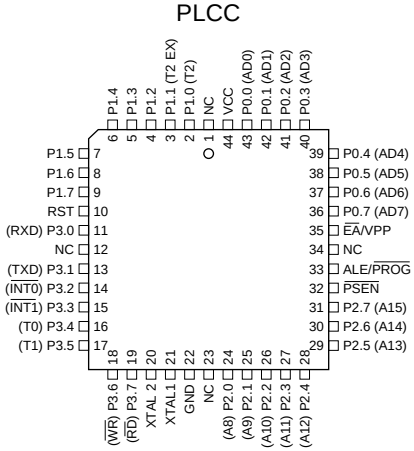
PQFP/TQFP



PDIP



PLCC



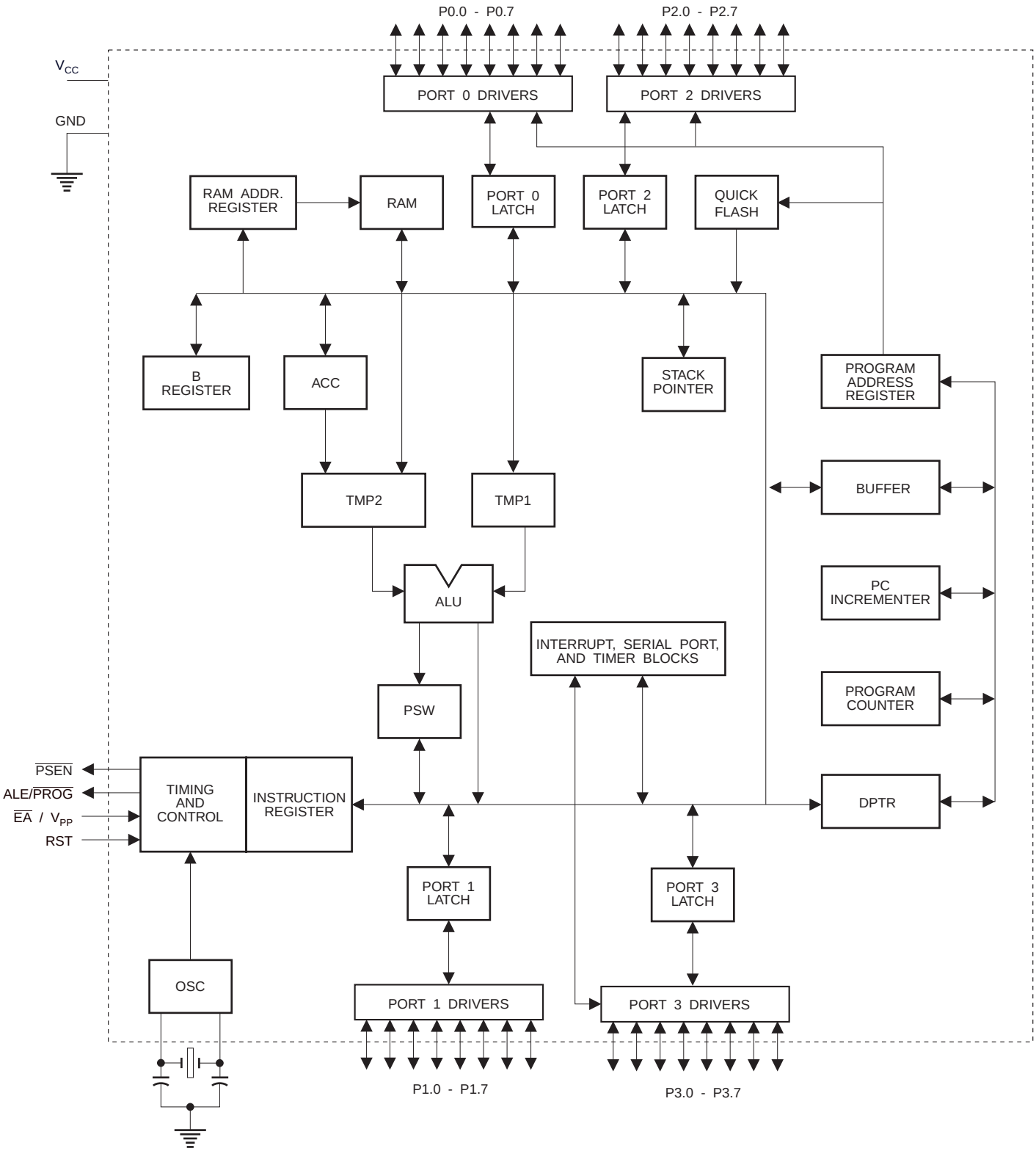
8-bit Microcontroller with 8K Bytes Flash

AT89C52

Not Recommended for New Designs. Use AT89S52.

Rev. 0313H–02/00

Block Diagram



The AT89C52 provides the following standard features: 8K bytes of Flash, 256 bytes of RAM, 32 I/O lines, three 16-bit timer/counters, a six-vector two-level interrupt architecture, a full-duplex serial port, on-chip oscillator, and clock circuitry. In addition, the AT89C52 is designed with static logic for operation down to zero frequency and supports two software selectable power saving modes. The Idle Mode stops the CPU while allowing the RAM, timer/counters, serial port, and interrupt system to continue functioning. The Power-down mode saves the RAM contents but freezes the oscillator, disabling all other chip functions until the next hardware reset.

Pin Description

VCC
Supply voltage.

GND
Ground.

Port 0

Port 0 is an 8-bit open drain bi-directional I/O port. As an output port, each pin can sink eight TTL inputs. When 1s are written to port 0 pins, the pins can be used as high-impedance inputs.

Port 0 can also be configured to be the multiplexed low-order address/data bus during accesses to external program and data memory. In this mode, P0 has internal pullups.

Port 0 also receives the code bytes during Flash programming and outputs the code bytes during program verification. External pullups are required during program verification.

Port 1

Port 1 is an 8-bit bi-directional I/O port with internal pullups. The Port 1 output buffers can sink/source four TTL inputs. When 1s are written to Port 1 pins, they are pulled high by the internal pullups and can be used as inputs. As inputs, Port 1 pins that are externally being pulled low will source current (I_{IL}) because of the internal pullups.

In addition, P1.0 and P1.1 can be configured to be the timer/counter 2 external count input (P1.0/T2) and the timer/counter 2 trigger input (P1.1/T2EX), respectively, as shown in the following table.
Port 1 also receives the low-order address bytes during Flash programming and verification.

Port Pin	Alternate Functions
P1.0	T2 (external count input to Timer/Counter 2), clock-out
P1.1	T2EX (Timer/Counter 2 capture/reload trigger and direction control)

Port 2

Port 2 is an 8-bit bi-directional I/O port with internal pullups. The Port 2 output buffers can sink/source four TTL inputs. When 1s are written to Port 2 pins, they are pulled high by the internal pullups and can be used as inputs. As inputs, Port 2 pins that are externally being pulled low will source current (I_{IL}) because of the internal pullups.

Port 2 emits the high-order address byte during fetches from external program memory and during accesses to external data memory that use 16-bit addresses (MOVX @ DPTR). In this application, Port 2 uses strong internal pullups when emitting 1s. During accesses to external data memory that use 8-bit addresses (MOVX @ RI), Port 2 emits the contents of the P2 Special Function Register.

Port 2 also receives the high-order address bits and some control signals during Flash programming and verification.

Port 3

Port 3 is an 8-bit bi-directional I/O port with internal pullups. The Port 3 output buffers can sink/source four TTL inputs. When 1s are written to Port 3 pins, they are pulled high by the internal pullups and can be used as inputs. As inputs, Port 3 pins that are externally being pulled low will source current (I_{IL}) because of the pullups.

Port 3 also serves the functions of various special features of the AT89C51, as shown in the following table.

Port 3 also receives some control signals for Flash programming and verification.

Port Pin	Alternate Functions
P3.0	RXD (serial input port)
P3.1	TXD (serial output port)
P3.2	$\overline{INT0}$ (external interrupt 0)
P3.3	$\overline{INT1}$ (external interrupt 1)
P3.4	T0 (timer 0 external input)
P3.5	T1 (timer 1 external input)
P3.6	$\overline{WR}$ (external data memory write strobe)
P3.7	$\overline{RD}$ (external data memory read strobe)

RST
Reset input. A high on this pin for two machine cycles while the oscillator is running resets the device.

ALE/$\overline{PROG}$
Address Latch Enable is an output pulse for latching the low byte of the address during accesses to external memory. This pin is also the program pulse input ($\overline{PROG}$) during Flash programming.

In normal operation, ALE is emitted at a constant rate of 1/6 the oscillator frequency and may be used for external

timing or clocking purposes. Note, however, that one ALE pulse is skipped during each access to external data memory.

If desired, ALE operation can be disabled by setting bit 0 of SFR location 8EH. With the bit set, ALE is active only during a MOVX or MOVC instruction. Otherwise, the pin is weakly pulled high. Setting the ALE-disable bit has no effect if the microcontroller is in external execution mode.

$\overline{\text{PSEN}}$

Program Store Enable is the read strobe to external program memory.

When the AT89C52 is executing code from external program memory, $\overline{\text{PSEN}}$ is activated twice each machine cycle, except that two $\overline{\text{PSEN}}$ activations are skipped during each access to external data memory.

$\overline{\text{EA}}$/VPP

External Access Enable. $\overline{\text{EA}}$ must be strapped to GND in order to enable the device to fetch code from external program memory locations starting at 0000H up to FFFFH. Note, however, that if lock bit 1 is programmed, $\overline{\text{EA}}$ will be internally latched on reset.

$\overline{\text{EA}}$ should be strapped to V_{CC} for internal program executions.

This pin also receives the 12-volt programming enable voltage (V_{PP}) during Flash programming when 12-volt programming is selected.

XTAL1

Input to the inverting oscillator amplifier and input to the internal clock operating circuit.

XTAL2

Output from the inverting oscillator amplifier.

Table 1. AT89C52 SFR Map and Reset Values

0F8H									0FFH
0F0H	B 00000000								0F7H
0E8H									0EFH
0E0H	ACC 00000000								0E7H
0D8H									0DFH
0D0H	PSW 00000000								0D7H
0C8H	T2CON 00000000	T2MOD XXXXXX00	RCAP2L 00000000	RCAP2H 00000000	TL2 00000000	TH2 00000000			0CFH
0C0H									0C7H
0B8H	IP XX000000								0BFH
0B0H	P3 11111111								0B7H
0A8H	IE 0X000000								0AFH
0A0H	P2 11111111								0A7H
98H	SCON 00000000	SBUF XXXXXXXX							9FH
90H	P1 11111111								97H
88H	TCON 00000000	TMOD 00000000	TL0 00000000	TL1 00000000	TH0 00000000	TH1 00000000			8FH
80H	P0 11111111	SP 00000111	DPL 00000000	DPH 00000000				PCON 0XXX0000	87H

Special Function Registers

A map of the on-chip memory area called the Special Function Register (SFR) space is shown in Table 1.

Note that not all of the addresses are occupied, and unoccupied addresses may not be implemented on the chip. Read accesses to these addresses will in general return random data, and write accesses will have an indeterminate effect.

User software should not write 1s to these unlisted locations, since they may be used in future products to invoke new features. In that case, the reset or inactive values of the new bits will always be 0.

Timer 2 Registers Control and status bits are contained in registers T2CON (shown in Table 2) and T2MOD (shown in Table 4) for Timer 2. The register pair (RCAP2H, RCAP2L) are the Capture/Reload registers for Timer 2 in 16-bit capture mode or 16-bit auto-reload mode.

Interrupt Registers The individual interrupt enable bits are in the IE register. Two priorities can be set for each of the six interrupt sources in the IP register.r

Table 2. T2CON – Timer/Counter 2 Control Registe

T2CON Address = 0C8H — Reset Value = 0000 0000B

Bit Addressable

Bit	TF2	EXF2	RCLK	TCLK	EXEN2	TR2	C/$\overline{T2}$	CP/$\overline{RL2}$
	7	6	5	4	3	2	1	0

Symbol	Function
TF2	Timer 2 overflow flag set by a Timer 2 overflow and must be cleared by software. TF2 will not be set when either RCLK = 1 or TCLK = 1.
EXF2	Timer 2 external flag set when either a capture or reload is caused by a negative transition on T2EX and EXEN2 = 1. When Timer 2 interrupt is enabled, EXF2 = 1 will cause the CPU to vector to the Timer 2 interrupt routine. EXF2 must be cleared by software. EXF2 does not cause an interrupt in up/down counter mode (DCEN = 1).
RCLK	Receive clock enable. When set, causes the serial port to use Timer 2 overflow pulses for its receive clock in serial port Modes 1 and 3. RCLK = 0 causes Timer 1 overflow to be used for the receive clock.
TCLK	Transmit clock enable. When set, causes the serial port to use Timer 2 overflow pulses for its transmit clock in serial port Modes 1 and 3. TCLK = 0 causes Timer 1 overflows to be used for the transmit clock.
EXEN2	Timer 2 external enable. When set, allows a capture or reload to occur as a result of a negative transition on T2EX if Timer 2 is not being used to clock the serial port. EXEN2 = 0 causes Timer 2 to ignore events at T2EX.
TR2	Start/Stop control for Timer 2. TR2 = 1 starts the timer.
C/$\overline{T2}$	Timer or counter select for Timer 2. C/$\overline{T2}$ = 0 for timer function. C/$\overline{T2}$ = 1 for external event counter (falling edge triggered).
CP/$\overline{RL2}$	Capture/Reload select. CP/$\overline{RL2}$ = 1 causes captures to occur on negative transitions at T2EX if EXEN2 = 1. CP/$\overline{RL2}$ = 0 causes automatic reloads to occur when Timer 2 overflows or negative transitions occur at T2EX when EXEN2 = 1. When either RCLK or TCLK = 1, this bit is ignored and the timer is forced to auto-reload on Timer 2 overflow.

Data Memory

The AT89C52 implements 256 bytes of on-chip RAM. The upper 128 bytes occupy a parallel address space to the Special Function Registers. That means the upper 128 bytes have the same addresses as the SFR space but are physically separate from SFR space.

When an instruction accesses an internal location above address 7FH, the address mode used in the instruction specifies whether the CPU accesses the upper 128 bytes of RAM or the SFR space. Instructions that use direct addressing access SFR space.

For example, the following direct addressing instruction accesses the SFR at location 0A0H (which is P2).

```
MOV 0A0H, #data
```

Instructions that use indirect addressing access the upper 128 bytes of RAM. For example, the following indirect addressing instruction, where R0 contains 0A0H, accesses the data byte at address 0A0H, rather than P2 (whose address is 0A0H).

```
MOV @R0, #data
```

Note that stack operations are examples of indirect addressing, so the upper 128 bytes of data RAM are available as stack space.

Timer 0 and 1

Timer 0 and Timer 1 in the AT89C52 operate the same way as Timer 0 and Timer 1 in the AT89C51.

Timer 2

Timer 2 is a 16-bit Timer/Counter that can operate as either a timer or an event counter. The type of operation is selected by bit C/$\overline{T2}$ in the SFR T2CON (shown in Table 2). Timer 2 has three operating modes: capture, auto-reload (up or down counting), and baud rate generator. The modes are selected by bits in T2CON, as shown in Table 3.

Timer 2 consists of two 8-bit registers, TH2 and TL2. In the Timer function, the TL2 register is incremented every machine cycle. Since a machine cycle consists of 12 oscillator periods, the count rate is 1/12 of the oscillator frequency.

Table 3. Timer 2 Operating Modes

RCLK +TCLK	CP/$\overline{RL2}$	TR2	MODE
0	0	1	16-bit Auto-reload
0	1	1	16-bit Capture
1	X	1	Baud Rate Generator
X	X	0	(Off)

In the Counter function, the register is incremented in response to a 1-to-0 transition at its corresponding external input pin, T2. In this function, the external input is sampled during S5P2 of every machine cycle. When the samples show a high in one cycle and a low in the next cycle, the count is incremented. The new count value appears in the register during S3P1 of the cycle following the one in which the transition was detected. Since two machine cycles (24 oscillator periods) are required to recognize a 1-to-0 transition, the maximum count rate is 1/24 of the oscillator frequency. To ensure that a given level is sampled at least once before it changes, the level should be held for at least one full machine cycle.

Capture Mode

In the capture mode, two options are selected by bit EXEN2 in T2CON. If EXEN2 = 0, Timer 2 is a 16-bit timer or counter which upon overflow sets bit TF2 in T2CON. This bit can then be used to generate an interrupt. If EXEN2 = 1, Timer 2 performs the same operation, but a 1-to-0 transition at external input T2EX also causes the current value in TH2 and TL2 to be captured into RCAP2H and RCAP2L, respectively. In addition, the transition at T2EX causes bit EXF2 in T2CON to be set. The EXF2 bit, like TF2, can generate an interrupt. The capture mode is illustrated in Figure 1.

Auto-reload (Up or Down Counter)

Timer 2 can be programmed to count up or down when configured in its 16-bit auto-reload mode. This feature is invoked by the DCEN (Down Counter Enable) bit located in the SFR T2MOD (see Table 4). Upon reset, the DCEN bit is set to 0 so that timer 2 will default to count up. When DCEN is set, Timer 2 can count up or down, depending on the value of the T2EX pin.

Figure 1. Timer in Capture Mode

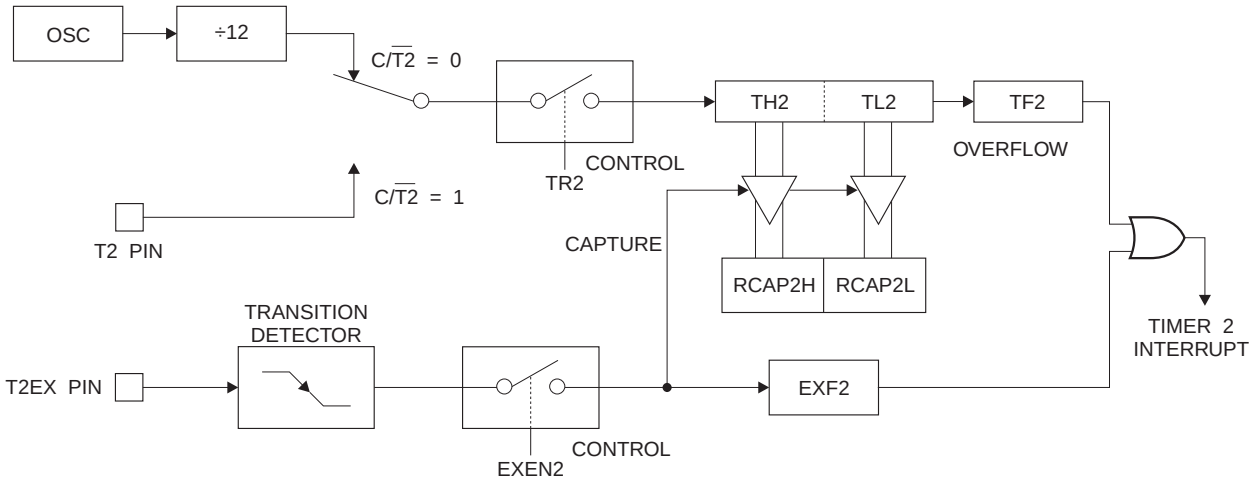


Figure 2 shows Timer 2 automatically counting up when DCEN = 0. In this mode, two options are selected by bit EXEN2 in T2CON. If EXEN2 = 0, Timer 2 counts up to 0FFFFH and then sets the TF2 bit upon overflow. The overflow also causes the timer registers to be reloaded with the 16-bit value in RCAP2H and RCAP2L. The values in Timer in Capture ModeRCAP2H and RCAP2L are preset by software. If EXEN2 = 1, a 16-bit reload can be triggered either by an overflow or by a 1-to-0 transition at external input T2EX. This transition also sets the EXF2 bit. Both the TF2 and EXF2 bits can generate an interrupt if enabled.

Setting the DCEN bit enables Timer 2 to count up or down, as shown in Figure 3. In this mode, the T2EX pin controls the direction of the count. A logic 1 at T2EX makes Timer 2 count up. The timer will overflow at 0FFFFH and set the TF2 bit. This overflow also causes the 16-bit value in RCAP2H and RCAP2L to be reloaded into the timer registers, TH2 and TL2, respectively.

A logic 0 at T2EX makes Timer 2 count down. The timer underflows when TH2 and TL2 equal the values stored in RCAP2H and RCAP2L. The underflow sets the TF2 bit and causes 0FFFFH to be reloaded into the timer registers.

The EXF2 bit toggles whenever Timer 2 overflows or underflows and can be used as a 17th bit of resolution. In this operating mode, EXF2 does not flag an interrupt.

Figure 2. Timer 2 Auto Reload Mode (DCEN = 0)

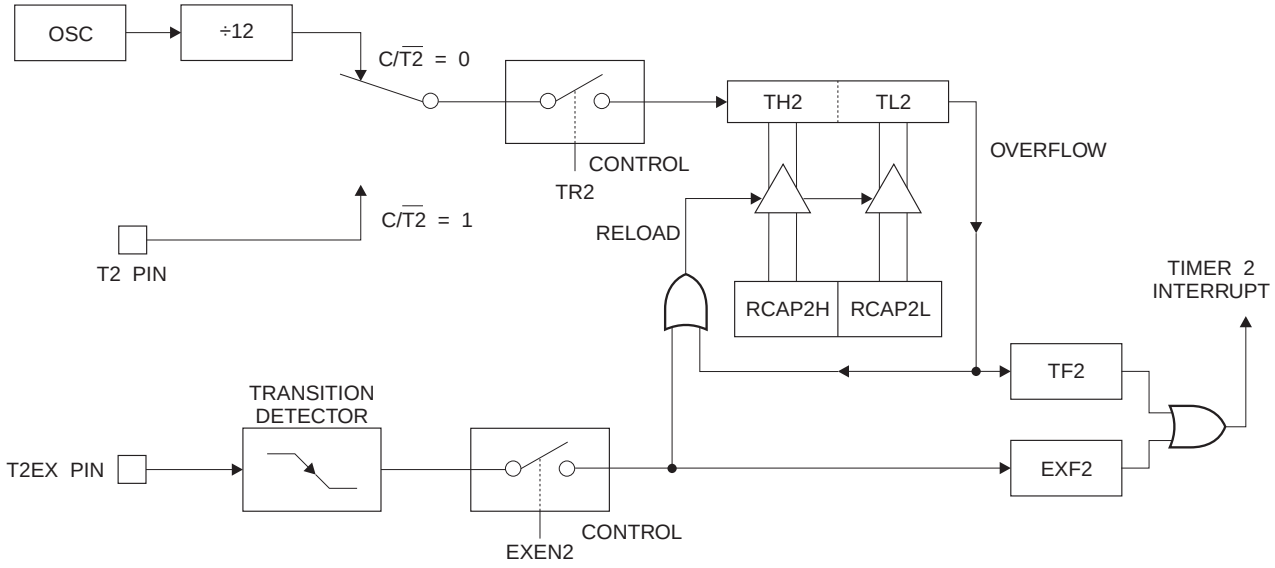


Table 4. T2MOD – Timer 2 Mode Control Register

T2MOD Address = 0C9H — Reset Value = XXXX XX00B

Not Bit Addressable

	–	–	–	–	–	–	T2OE	DCEN
Bit	7	6	5	4	3	2	1	0

Symbol	Function
–	Not implemented, reserved for future
T2OE	Timer 2 Output Enable bit.
DCEN	When set, this bit allows Timer 2 to be configured as an up/down counter.

Figure 3. Timer 2 Auto Reload Mode (DCEN = 1)

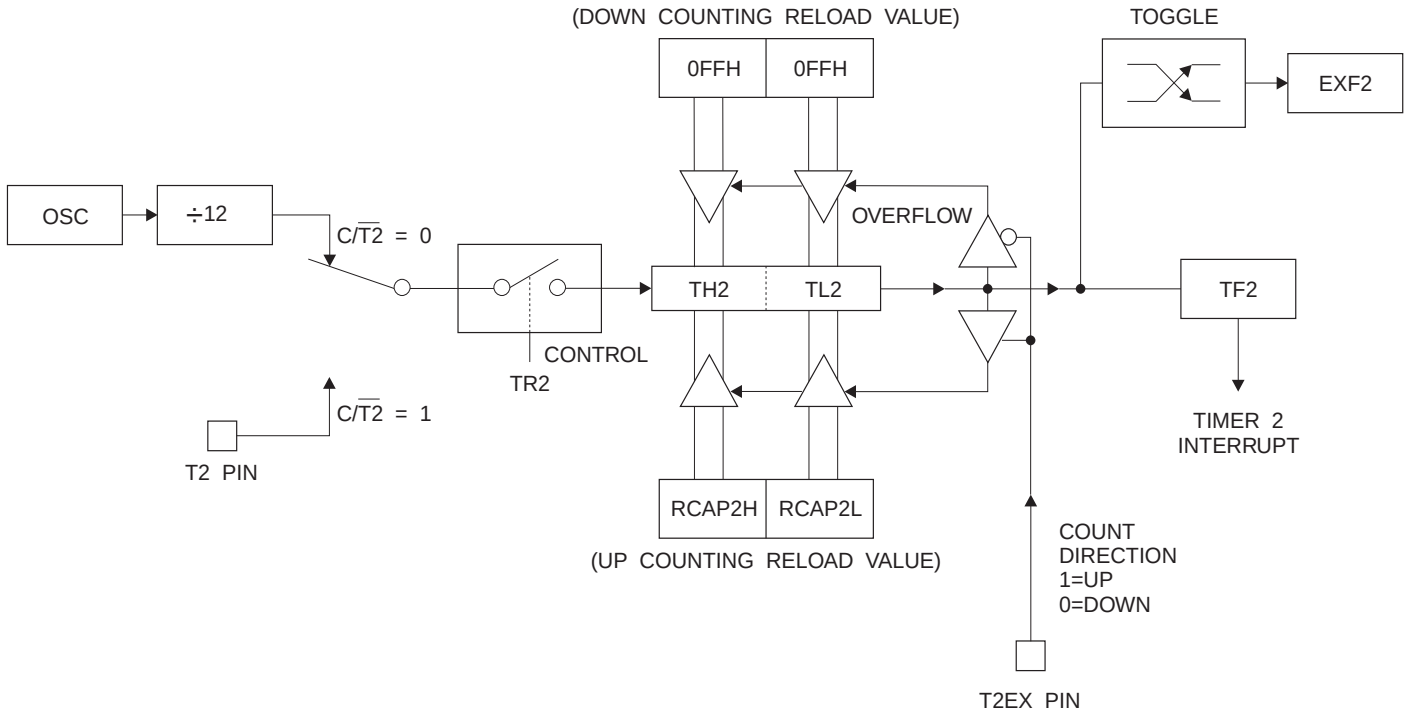
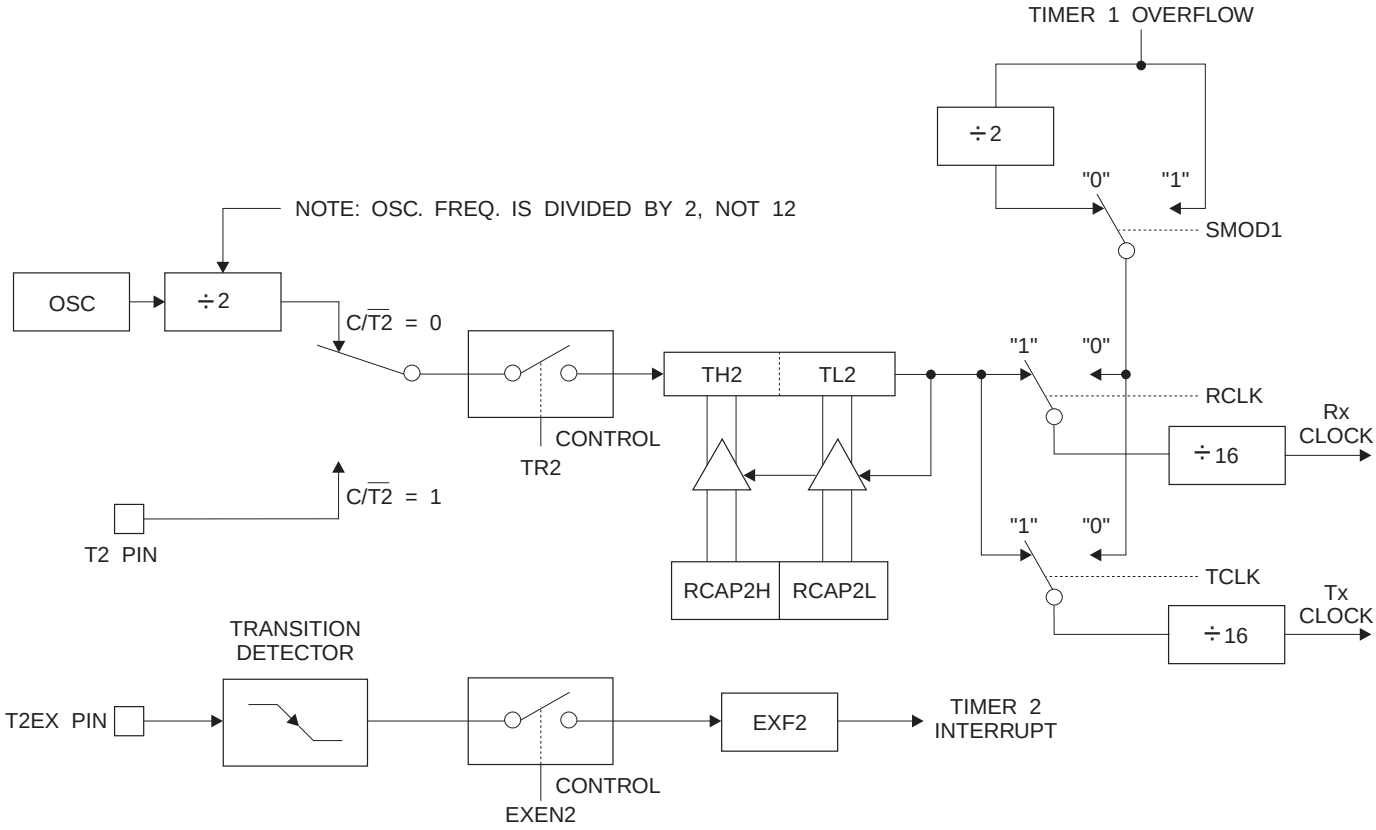


Figure 4. Timer 2 in Baud Rate Generator Mode



Baud Rate Generator

Timer 2 is selected as the baud rate generator by setting TCLK and/or RCLK in T2CON (Table 2). Note that the baud rates for transmit and receive can be different if Timer 2 is used for the receiver or transmitter and Timer 1 is used for the other function. Setting RCLK and/or TCLK puts Timer 2 into its baud rate generator mode, as shown in Figure 4.

The baud rate generator mode is similar to the auto-reload mode, in that a rollover in TH2 causes the Timer 2 registers to be reloaded with the 16-bit value in registers RCAP2H and RCAP2L, which are preset by software.

The baud rates in Modes 1 and 3 are determined by Timer 2's overflow rate according to the following equation.

$$\text{Modes 1 and 3 Baud Rates} = \frac{\text{Timer 2 Overflow Rate}}{16}$$

The Timer can be configured for either timer or counter operation. In most applications, it is configured for timer operation (CP/$\overline{T2}$ = 0). The timer operation is different for Timer 2 when it is used as a baud rate generator. Normally, as a timer, it increments every machine cycle (at 1/12 the oscillator frequency). As a baud rate generator, however, it increments every state time (at 1/2 the oscillator frequency). The baud rate formula is given below.

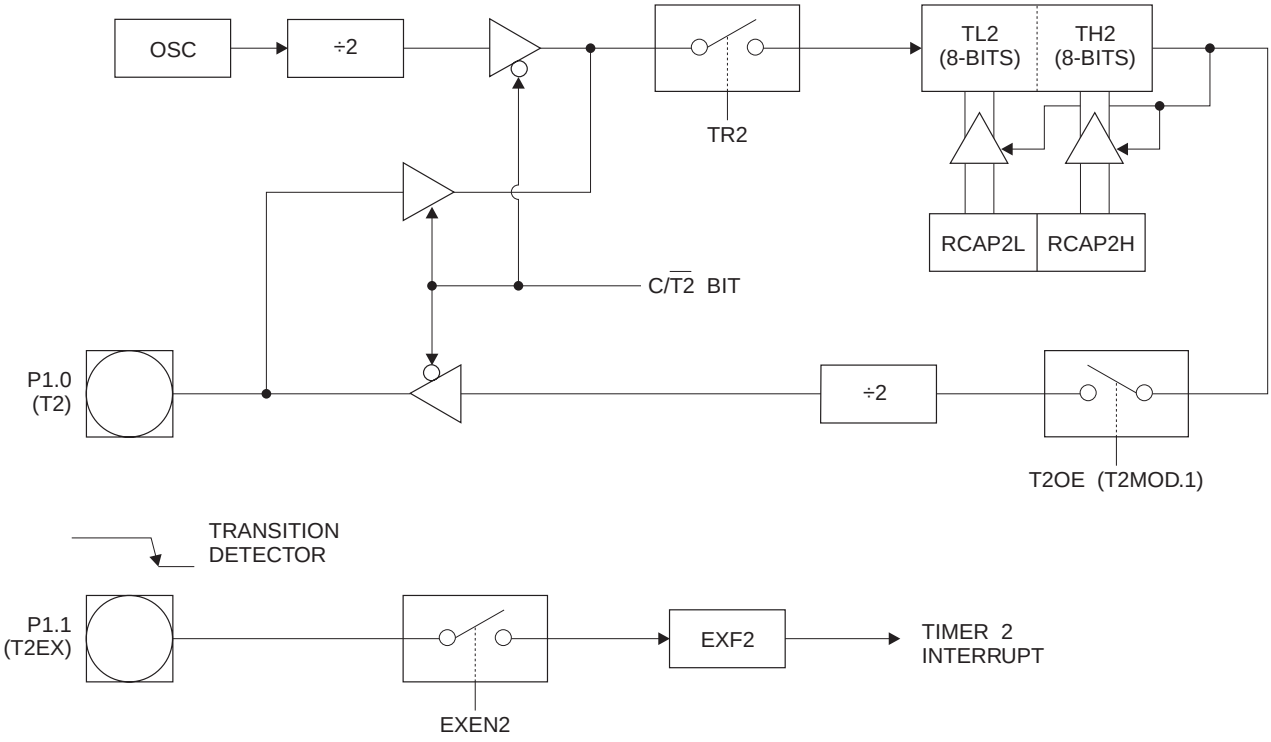
$$\frac{\text{Modes 1 and 3}}{\text{Baud Rate}} = \frac{\text{Oscillator Frequency}}{32 \times [65536 - (\text{RCAP2H,RCAP2L})]}$$

where (RCAP2H, RCAP2L) is the content of RCAP2H and RCAP2L taken as a 16-bit unsigned integer.

Timer 2 as a baud rate generator is shown in Figure 4. This figure is valid only if RCLK or TCLK = 1 in T2CON. Note that a rollover in TH2 does not set TF2 and will not generate an interrupt. Note too, that if EXEN2 is set, a 1-to-0 transition in T2EX will set EXF2 but will not cause a reload from (RCAP2H, RCAP2L) to (TH2, TL2). Thus when Timer 2 is in use as a baud rate generator, T2EX can be used as an extra external interrupt.

Note that when Timer 2 is running (TR2 = 1) as a timer in the baud rate generator mode, TH2 or TL2 should not be read from or written to. Under these conditions, the Timer is incremented every state time, and the results of a read or write may not be accurate. The RCAP2 registers may be read but should not be written to, because a write might overlap a reload and cause write and/or reload errors. The timer should be turned off (clear TR2) before accessing the Timer 2 or RCAP2 registers.

Figure 5. Timer 2 in Clock-out Mode



Programmable Clock Out

A 50% duty cycle clock can be programmed to come out on P1.0, as shown in Figure 5. This pin, besides being a regular I/O pin, has two alternate functions. It can be programmed to input the external clock for Timer/Counter 2 or to output a 50% duty cycle clock ranging from 61 Hz to 4 MHz at a 16 MHz operating frequency.

To configure the Timer/Counter 2 as a clock generator, bit C/$\overline{T2}$ (T2CON.1) must be cleared and bit T2OE (T2MOD.1) must be set. Bit TR2 (T2CON.2) starts and stops the timer.

The clock-out frequency depends on the oscillator frequency and the reload value of Timer 2 capture registers (RCAP2H, RCAP2L), as shown in the following equation.

$$\text{Clock-Out Frequency} = \frac{\text{Oscillator Fequency}}{4 \times [65536 - (\text{RCAP2H,RCAP2L})]}$$

In the clock-out mode, Timer 2 roll-overs will not generate an interrupt. This behavior is similar to when Timer 2 is used as a baud-rate generator. It is possible to use Timer 2 as a baud-rate generator and a clock generator simultaneously. Note, however, that the baud-rate and clock-out frequencies cannot be determined independently from one another since they both use RCAP2H and RCAP2L.

UART

The UART in the AT89C52 operates the same way as the UART in the AT89C51.

Interrupts

The AT89C52 has a total of six interrupt vectors: two external interrupts ($\overline{INT0}$ and $\overline{INT1}$), three timer interrupts (Timers 0, 1, and 2), and the serial port interrupt. These interrupts are all shown in Figure 6.

Each of these interrupt sources can be individually enabled or disabled by setting or clearing a bit in Special Function Register IE. IE also contains a global disable bit, EA, which disables all interrupts at once.

Note that Table shows that bit position IE.6 is unimplemented. In the AT89C51, bit position IE.5 is also unimplemented. User software should not write 1s to these bit positions, since they may be used in future AT89 products.

Timer 2 interrupt is generated by the logical OR of bits TF2 and EXF2 in register T2CON. Neither of these flags is cleared by hardware when the service routine is vectored to. In fact, the service routine may have to determine whether it was TF2 or EXF2 that generated the interrupt, and that bit will have to be cleared in software.

The Timer 0 and Timer 1 flags, TF0 and TF1, are set at S5P2 of the cycle in which the timers overflow. The values are then polled by the circuitry in the next cycle. However, the Timer 2 flag, TF2, is set at S2P2 and is polled in the same cycle in which the timer overflows.

Table 5. Interrupt Enable (IE) Register

(MSB) (LSB)

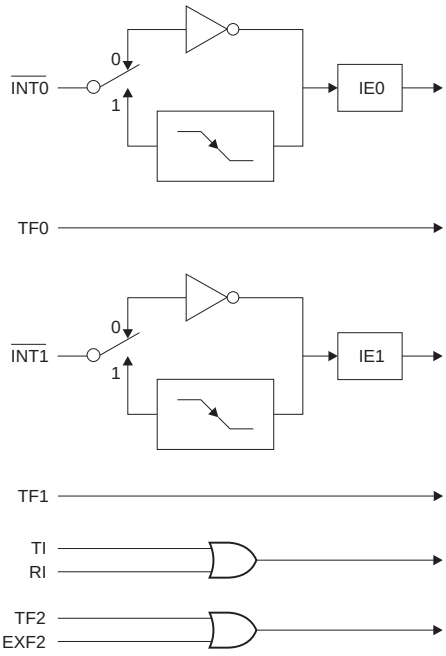
EA	–	ET2	ES	ET1	EX1	ET0	EX0

Enable Bit = 1 enables the interrupt.

Enable Bit = 0 disables the interrupt.

Symbol	Position	Function
EA	IE.7	Disables all interrupts. If EA = 0, no interrupt is acknowledged. If EA = 1, each interrupt source is individually enabled or disabled by setting or clearing its enable bit.
–	IE.6	Reserved.
ET2	IE.5	Timer 2 interrupt enable bit.
ES	IE.4	Serial Port interrupt enable bit.
ET1	IE.3	Timer 1 interrupt enable bit.
EX1	IE.2	External interrupt 1 enable bit.
ET0	IE.1	Timer 0 interrupt enable bit.
EX0	IE.0	External interrupt 0 enable bit.
User software should never write 1s to unimplemented bits, because they may be used in future AT89 products.		

Figure 6. Interrupt Sources



Oscillator Characteristics

XTAL1 and XTAL2 are the input and output, respectively, of an inverting amplifier that can be configured for use as an on-chip oscillator, as shown in Figure 7. Either a quartz crystal or ceramic resonator may be used. To drive the device from an external clock source, XTAL2 should be left unconnected while XTAL1 is driven, as shown in Figure 8. There are no requirements on the duty cycle of the external clock signal, since the input to the internal clocking circuitry is through a divide-by-two flip-flop, but minimum and maximum voltage high and low time specifications must be observed.

Idle Mode

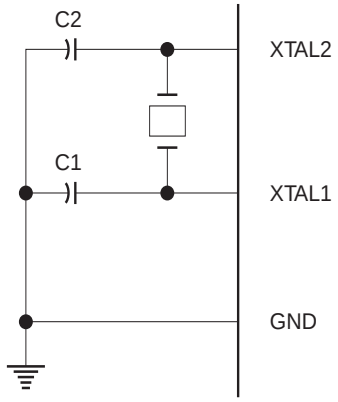
In idle mode, the CPU puts itself to sleep while all the on-chip peripherals remain active. The mode is invoked by software. The content of the on-chip RAM and all the special functions registers remain unchanged during this mode. The idle mode can be terminated by any enabled interrupt or by a hardware reset.

Note that when idle mode is terminated by a hardware reset, the device normally resumes program execution from where it left off, up to two machine cycles before the internal reset algorithm takes control. On-chip hardware inhibits access to internal RAM in this event, but access to the port pins is not inhibited. To eliminate the possibility of an unexpected write to a port pin when idle mode is terminated by a reset, the instruction following the one that invokes idle mode should not write to a port pin or to external memory.

Power-down Mode

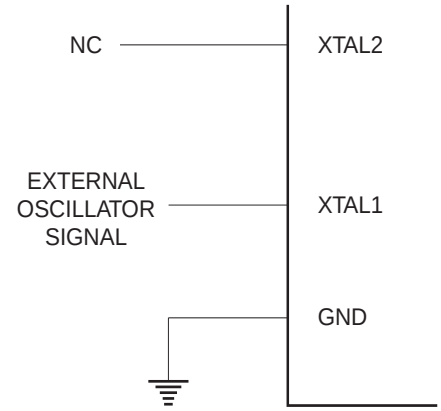
In the power-down mode, the oscillator is stopped, and the instruction that invokes power-down is the last instruction executed. The on-chip RAM and Special Function Registers retain their values until the power-down mode is terminated. The only exit from power-down is a hardware reset. Reset redefines the SFRs but does not change the on-chip RAM. The reset should not be activated before V_{CC} is restored to its normal operating level and must be held active long enough to allow the oscillator to restart and stabilize.

Figure 7. Oscillator Connections



Note: C1, C2 = 30 pF ±10 pF for Crystals
= 40 pF ±10 pF for Ceramic Resonators

Figure 8. External Clock Drive Configuration



Status of External Pins During Idle and Powe-down Modes

Mode	Program Memory	ALE	$\overline{PSEN}$	PORT0	PORT1	PORT2	PORT3
Idle	Internal	1	1	Data	Data	Data	Data
Idle	External	1	1	Float	Data	Address	Data
Power-down	Internal	0	0	Data	Data	Data	Data
Power-down	External	0	0	Float	Data	Data	Data

Program Memory Lock Bits

The AT89C52 has three lock bits that can be left unprogrammed (U) or can be programmed (P) to obtain the additional features listed in the following table.

Lock Bit Protection Modes

Program Lock Bits				
	LB1	LB2	LB3	Protection Type
1	U	U	U	No program lock features.
2	P	U	U	MOVC instructions executed from external program memory are disabled from fetching code bytes from internal memory, $\overline{EA}$ is sampled and latched on reset, and further programming of the Flash memory is disabled.
3	P	P	U	Same as mode 2, but verify is also disabled.
4	P	P	P	Same as mode 3, but external execution is also disabled.

When lock bit 1 is programmed, the logic level at the $\overline{EA}$ pin is sampled and latched during reset. If the device is powered up without a reset, the latch initializes to a random value and holds that value until reset is activated. The latched value of $\overline{EA}$ must agree with the current logic level at that pin in order for the device to function properly.

Programming the Flash

The AT89C52 is normally shipped with the on-chip Flash memory array in the erased state (that is, contents = FFH) and ready to be programmed. The programming interface accepts either a high-voltage (12-volt) or a low-voltage (V_{CC}) program enable signal. The Low-voltage programming mode provides a convenient way to program the AT89C52 inside the user's system, while the high-voltage programming mode is compatible with conventional third-party Flash or EPROM programmers.

The AT89C52 is shipped with either the high-voltage or low-voltage programming mode enabled. The respective top-side marking and device signature codes are listed in the following table.

	V_{PP} = 12V	V_{PP} = 5V
Top-side Mark	AT89C52 xxxx yyww	AT89C52 xxxx - 5 yyww
Signature	(030H) = 1EH (031H) = 52H (032H) = FFH	(030H) = 1EH (031H) = 52H (032H) = 05H

The AT89C52 code memory array is programmed byte-by-byte in either programming mode. *To program any non-blank byte in the on-chip Flash Memory, the entire memory must be erased using the Chip Erase Mode.*

Programming Algorithm Before programming the AT89C52, the address, data and control signals should be set up according to the Flash programming mode table and Figure 9 and Figure 10. To program the AT89C52, take the following steps.

1. Input the desired memory location on the address lines.
2. Input the appropriate data byte on the data lines.
3. Activate the correct combination of control signals.
4. Raise $\overline{EA}/V_{PP}$ to 12V for the high-voltage programming mode.
5. Pulse ALE/$\overline{PROG}$ once to program a byte in the Flash array or the lock bits. The byte-write cycle is self-timed and typically takes no more than 1.5 ms. Repeat steps 1 through 5, changing the address and data for the entire array or until the end of the object file is reached.

$\overline{Data}$ Polling The AT89C52 features $\overline{Data}$ Polling to indicate the end of a write cycle. During a write cycle, an attempted read of the last byte written will result in the complement of the written data on PO.7. Once the write cycle has been completed, true data is valid on all outputs, and the next cycle may begin. $\overline{Data}$ Polling may begin any time after a write cycle has been initiated.

Ready/$\overline{Busy}$ The progress of byte programming can also be monitored by the RDY/$\overline{BSY}$ output signal. P3.4 is pulled low after ALE goes high during programming to indicate $\overline{BUSY}$. P3.4 is pulled high again when programming is done to indicate READY.

Program Verify If lock bits LB1 and LB2 have not been programmed, the programmed code data can be read back via the address and data lines for verification. The lock bits cannot be verified directly. Verification of the lock bits is achieved by observing that their features are enabled.

Chip Erase The entire Flash array is erased electrically by using the proper combination of control signals and by holding ALE/$\overline{PROG}$ low for 10 ms. The code array is written with all 1s. The chip erase operation must be executed before the code memory can be reprogrammed.

Reading the Signature Bytes The signature bytes are read by the same procedure as a normal verification of locations 030H, 031H, and 032H, except that P3.6 and P3.7 must be pulled to a logic low. The values returned are as follows.

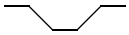
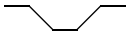
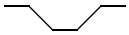
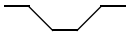
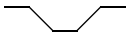
(030H) = 1EH indicates manufactured by Atmel
(031H) = 52H indicates 89C52
(032H) = FFH indicates 12V programming
(032H) = 05H indicates 5V programming

Programming Interface

Every code byte in the Flash array can be written, and the entire array can be erased, by using the appropriate combination of control signals. The write operation cycle is self-timed and once initiated, will automatically time itself to completion.

All major programming vendors offer worldwide support for the Atmel microcontroller series. Please contact your local programming vendor for the appropriate software revision.

Flash Programming Modes

Mode		RST	$\overline{PSEN}$	ALE/$\overline{PROG}$	$\overline{EA}$/V_{PP}	P2.6	P2.7	P3.6	P3.7
Write Code Data		H	L		H/12V	L	H	H	H
Read Code Data		H	L	H	H	L	L	H	H
Write Lock	Bit - 1	H	L		H/12V	H	H	H	H
	Bit - 2	H	L		H/12V	H	H	L	L
	Bit - 3	H	L		H/12V	H	L	H	L
Chip Erase		H	L	 (1)	H/12V	H	L	L	L
Read Signature Byte		H	L	H	H	L	L	L	L

Note: 1. Chip Erase requires a 10 ms $\overline{PROG}$ pulse.

Figure 9. Programming the Flash Memory

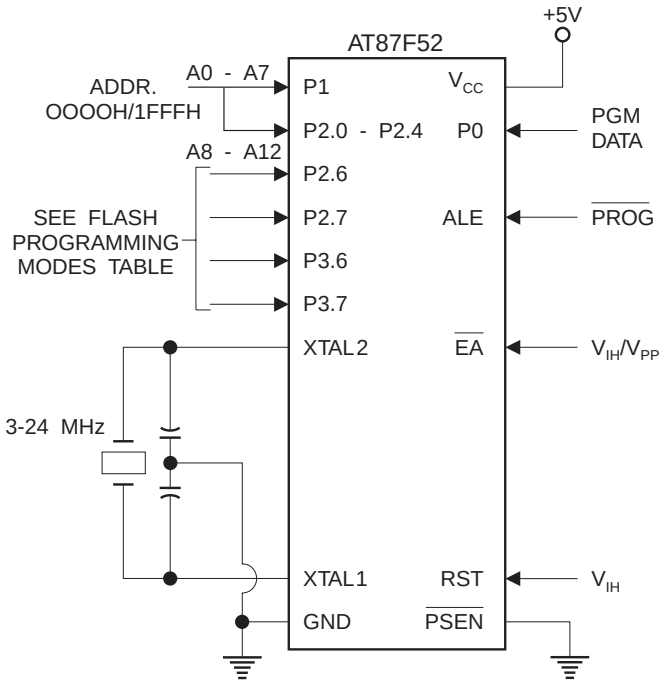
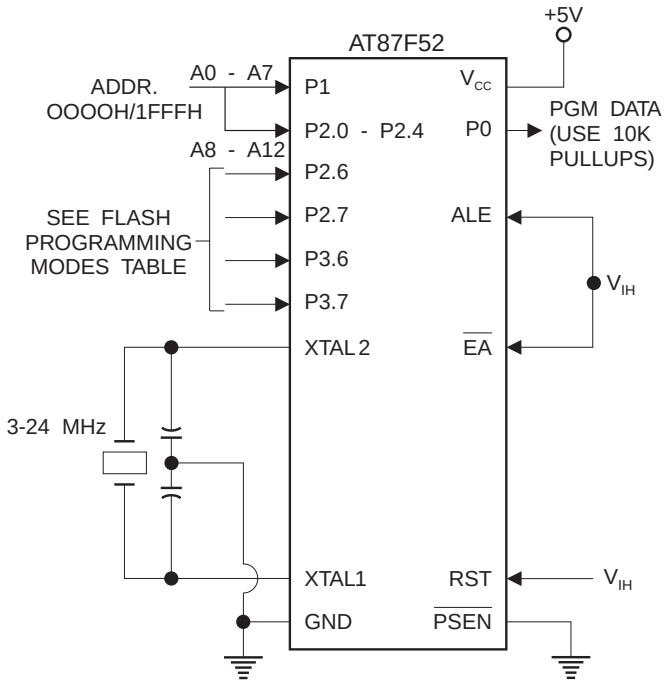


Figure 10. Verifying the Flash Memory



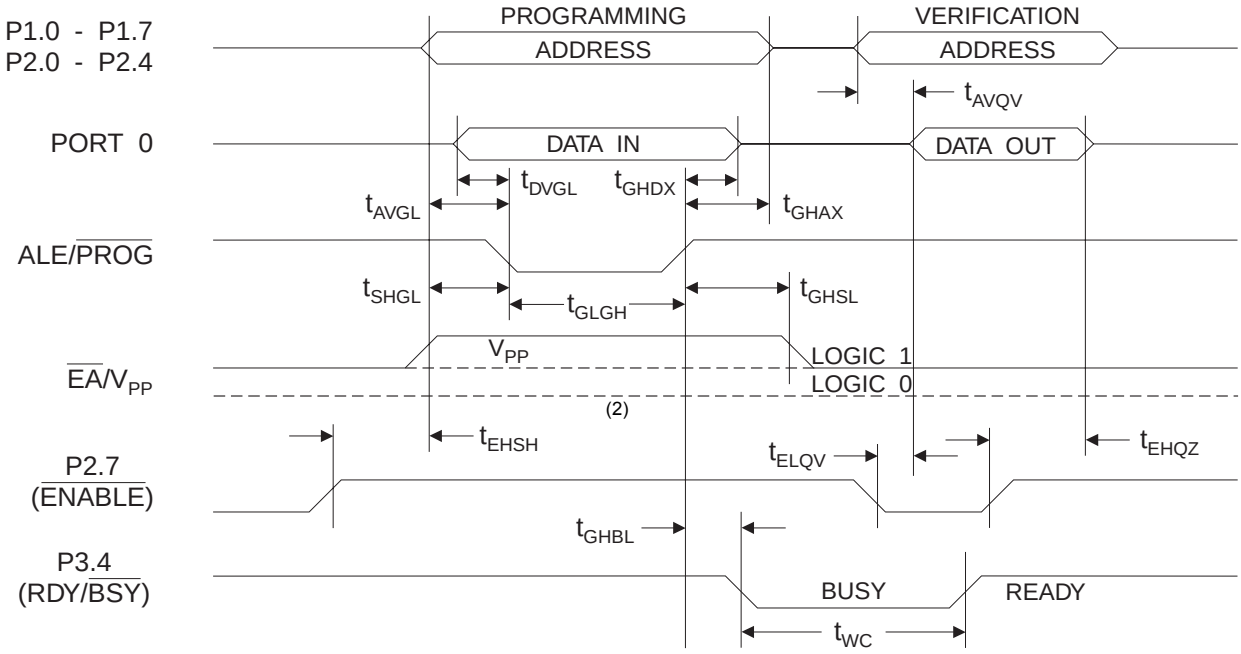
Flash Programming and Verification Characteristics

T_A = 0°C to 70°C, V_{CC} = 5.0 ±10%

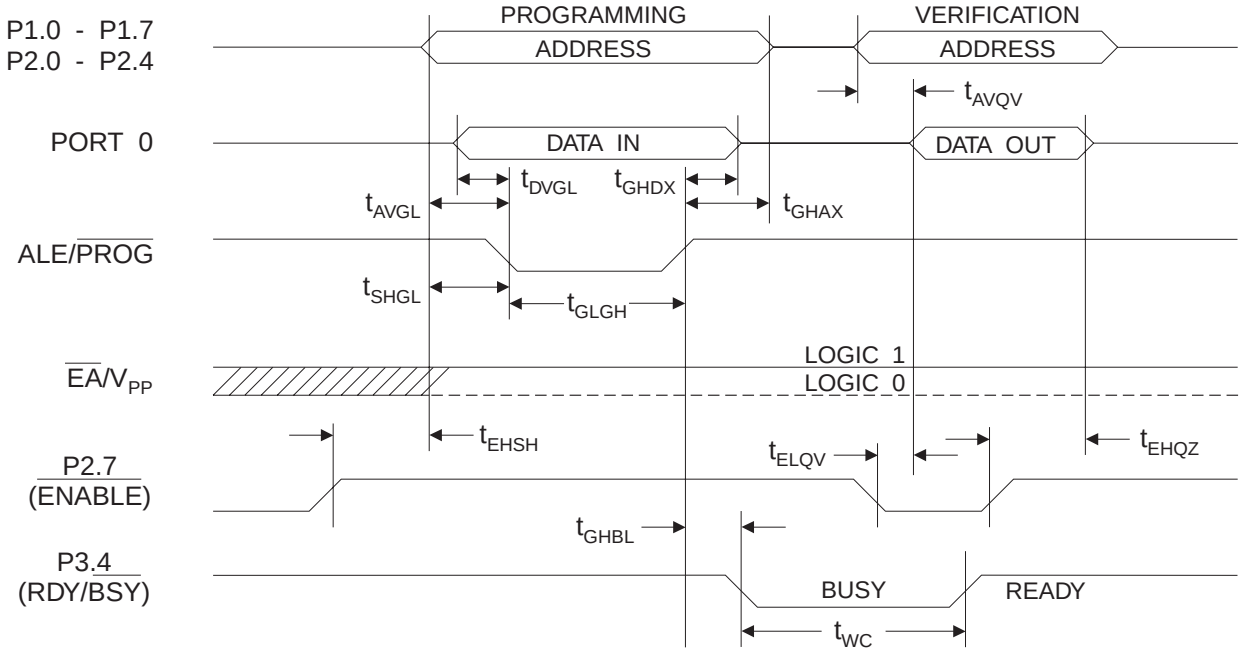
Symbol	Parameter	Min	Max	Units
V_{PP}(1)	Programming Enable Voltage	11.5	12.5	V
I_{PP}(1)	Programming Enable Current		1.0	mA
$1/t_{CLCL}$	Oscillator Frequency	3	24	MHz
t_{AVGL}	Address Setup to $\overline{PROG}$ Low	$48t_{CLCL}$		
t_{GHAX}	Address Hold after $\overline{PROG}$	$48t_{CLCL}$		
t_{DVGL}	Data Setup to $\overline{PROG}$ Low	$48t_{CLCL}$		
t_{GHDX}	Data Hold After $\overline{PROG}$	$48t_{CLCL}$		
t_{EHSH}	P2.7 ($\overline{ENABLE}$) High to V_{PP}	$48t_{CLCL}$		
t_{SHGL}	V_{PP} Setup to $\overline{PROG}$ Low	10		µs
t_{GHSL}(1)	V_{PP} Hold after $\overline{PROG}$	10		µs
t_{GLGH}	$\overline{PROG}$ Width	1	110	µs
t_{AVQV}	Address to Data Valid		$48t_{CLCL}$	
t_{ELQV}	$\overline{ENABLE}$ Low to Data Valid		$48t_{CLCL}$	
t_{EHQZ}	Data Float after $\overline{ENABLE}$	0	$48t_{CLCL}$	
t_{GHBL}	$\overline{PROG}$ High to $\overline{BUSY}$ Low		1.0	µs
t_{WC}	Byte Write Cycle Time		2.0	ms

Note: 1. Only used in 12-volt programming mode.

Flash Programming and Verification Waveforms - High-voltage Mode (V_{PP}=12V)



Flash Programming and Verification Waveforms - Low-voltage Mode (V_{PP}=5V)



Absolute Maximum Ratings*

Operating Temperature	-55°C to +125°C
Storage Temperature	-65°C to +150°C
Voltage on Any Pin with Respect to Ground	-1.0V to +7.0V
Maximum Operating Voltage	6.6V
DC Output Current	15.0 mA

*NOTICE: Stresses beyond those listed under "Absolute Maximum Ratings" may cause permanent damage to the device. This is a stress rating only and functional operation of the device at these or any other conditions beyond those indicated in the operational sections of this specification is not implied. Exposure to absolute maximum rating conditions for extended periods may affect device reliability.

DC Characteristics

The values shown in this table are valid for T_A = -40°C to 85°C and V_{CC} = 5.0V ±20%, unless otherwise noted.

Symbol	Parameter	Condition	Min	Max	Units
V_{IL}	Input Low-voltage	(Except $\overline{EA}$)	-0.5	0.2 V_{CC}-0.1	V
V_{IL1}	Input Low-voltage ($\overline{EA}$)		-0.5	0.2 V_{CC}-0.3	V
V_{IH}	Input High-voltage	(Except XTAL1, RST)	0.2 V_{CC}+0.9	V_{CC}+0.5	V
V_{IH1}	Input High-voltage	(XTAL1, RST)	0.7 V_{CC}	V_{CC}+0.5	V
V_{OL}	Output Low-voltage(1) (Ports 1,2,3)	I_{OL} = 1.6 mA		0.45	V
V_{OL1}	Output Low-voltage(1) (Port 0, ALE, $\overline{PSEN}$)	I_{OL} = 3.2 mA		0.45	V
V_{OH}	Output High-voltage (Ports 1,2,3, ALE, $\overline{PSEN}$)	I_{OH} = -60 µA, V_{CC} = 5V ±10%	2.4		V
		I_{OH} = -25 µA	0.75 V_{CC}		V
		I_{OH} = -10 µA	0.9 V_{CC}		V
V_{OH1}	Output High-voltage (Port 0 in External Bus Mode)	I_{OH} = -800 µA, V_{CC} = 5V ±10%	2.4		V
		I_{OH} = -300 µA	0.75 V_{CC}		V
		I_{OH} = -80 µA	0.9 V_{CC}		V
I_{IL}	Logical 0 Input Current (Ports 1,2,3)	V_{IN} = 0.45V		-50	µA
I_{TL}	Logical 1 to 0 Transition Current (Ports 1,2,3)	V_{IN} = 2V, V_{CC} = 5V ± 10%		-650	µA
I_{LI}	Input Leakage Current (Port 0, $\overline{EA}$)	0.45 < V_{IN} < V_{CC}		±10	µA
RRST	Reset Pulldown Resistor		50	300	KΩ
C_{IO}	Pin Capacitance	Test Freq. = 1 MHz, T_A = 25° C		10	pF
I_{CC}	Power Supply Current	Active Mode, 12 MHz		25	mA
		Idle Mode, 12 MHz		6.5	mA
	Power-down Mode(1)	V_{CC} = 6V		100	µA
		V_{CC} = 3V		40	µA

Notes:
1. Under steady state (non-transient) conditions, I_{OL} must be externally limited as follows:
   Maximum I_{OL} per port pin: 10 mA
   Maximum I_{OL} per 8-bit port:
   Port 0: 26 mA Ports 1, 2, 3: 15 mA
   Maximum total I_{OL} for all output pins: 71 mA
   If I_{OL} exceeds the test condition, V_{OL} may exceed the related specification. Pins are not guaranteed to sink current greater than the listed test conditions.
2. Minimum V_{CC} for Power-down is 2V.

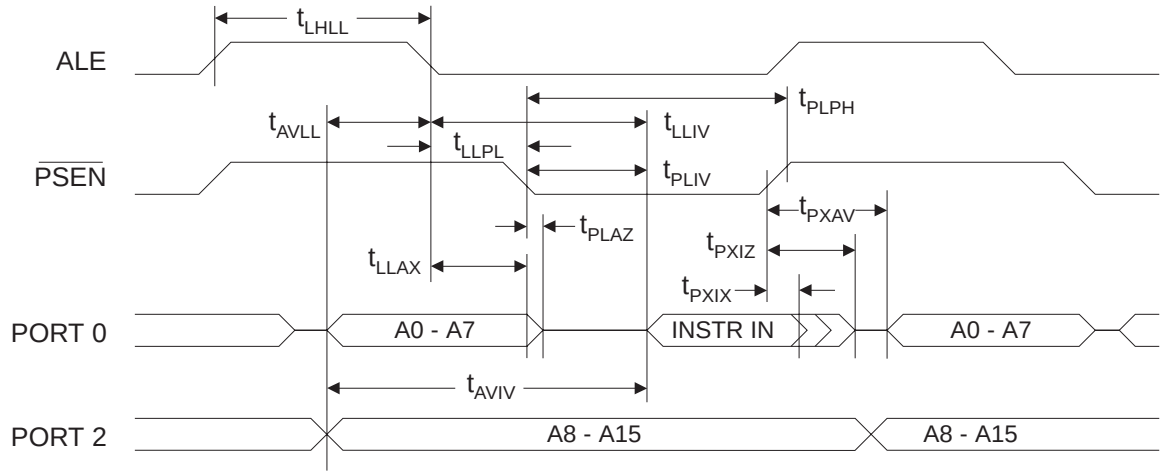
AC Characteristics

Under operating conditions, load capacitance for Port 0, ALE/$\overline{PROG}$, and $\overline{PSEN}$ = 100 pF; load capacitance for all other outputs = 80 pF.

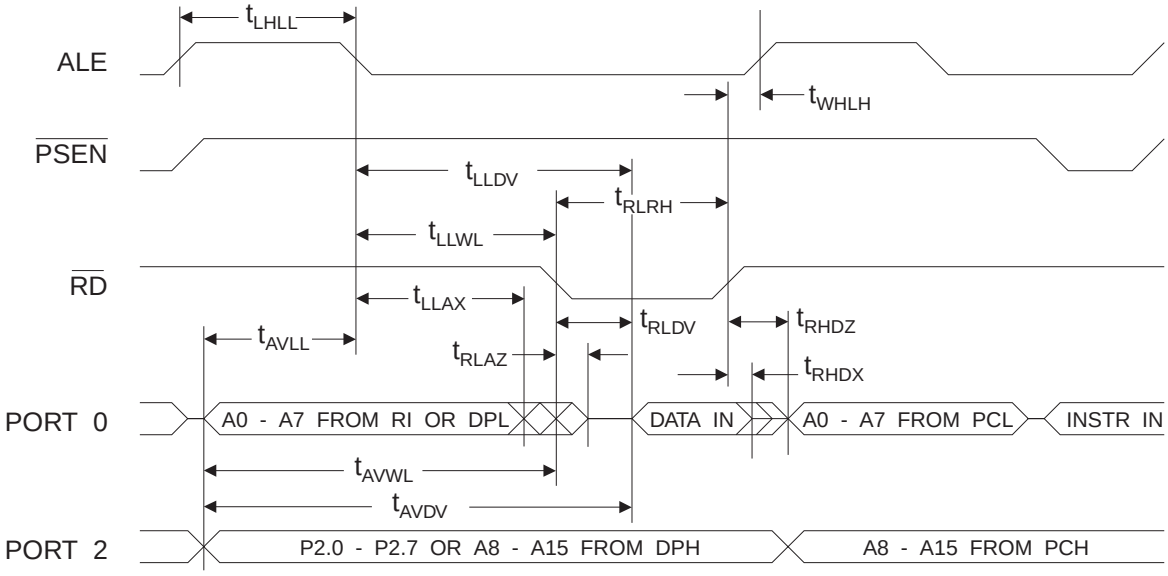
External Program and Data Memory Characteristics

Symbol	Parameter	12 MHz Oscillator		Variable Oscillator		Units
		Min	Max	Min	Max	
$1/t_{CLCL}$	Oscillator Frequency			0	24	MHz
t_{LHLL}	ALE Pulse Width	127		$2t_{CLCL}$-40		ns
t_{AVLL}	Address Valid to ALE Low	43		t_{CLCL}-13		ns
t_{LLAX}	Address Hold After ALE Low	48		t_{CLCL}-20		ns
t_{LLIV}	ALE Low to Valid Instruction In		233		$4t_{CLCL}$-65	ns
t_{LLPL}	ALE Low to $\overline{PSEN}$ Low	43		t_{CLCL}-13		ns
t_{PLPH}	$\overline{PSEN}$ Pulse Width	205		$3t_{CLCL}$-20		ns
t_{PLIV}	$\overline{PSEN}$ Low to Valid Instruction In		145		$3t_{CLCL}$-45	ns
t_{PXIX}	Input Instruction Hold after $\overline{PSEN}$	0		0		ns
t_{PXIZ}	Input Instruction Float after $\overline{PSEN}$		59		t_{CLCL}-10	ns
t_{PXAV}	$\overline{PSEN}$ to Address Valid	75		t_{CLCL}-8		ns
t_{AVIV}	Address to Valid Instruction In		312		$5t_{CLCL}$-55	ns
t_{PLAZ}	$\overline{PSEN}$ Low to Address Float		10		10	ns
t_{RLRH}	$\overline{RD}$ Pulse Width	400		$6t_{CLCL}$-100		ns
t_{WLWH}	$\overline{WR}$ Pulse Width	400		$6t_{CLCL}$-100		ns
t_{RLDV}	$\overline{RD}$ Low to Valid Data In		252		$5t_{CLCL}$-90	ns
t_{RHDX}	Data Hold After $\overline{RD}$	0		0		ns
t_{RHDZ}	Data Float After $\overline{RD}$		97		$2t_{CLCL}$-28	ns
t_{LLDV}	ALE Low to Valid Data In		517		$8t_{CLCL}$-150	ns
t_{AVDV}	Address to Valid Data In		585		$9t_{CLCL}$-165	ns
t_{LLWL}	ALE Low to $\overline{RD}$ or $\overline{WR}$ Low	200	300	$3t_{CLCL}$-50	$3t_{CLCL}$+50	ns
t_{AVWL}	Address to $\overline{RD}$ or $\overline{WR}$ Low	203		$4t_{CLCL}$-75		ns
t_{QVWX}	Data Valid to $\overline{WR}$ Transition	23		t_{CLCL}-20		ns
t_{QVWH}	Data Valid to $\overline{WR}$ High	433		$7t_{CLCL}$-120		ns
t_{WHQX}	Data Hold After $\overline{WR}$	33		t_{CLCL}-20		ns
t_{RLAZ}	$\overline{RD}$ Low to Address Float		0		0	ns
t_{WHLH}	$\overline{RD}$ or $\overline{WR}$ High to ALE High	43	123	t_{CLCL}-20	t_{CLCL}+25	ns

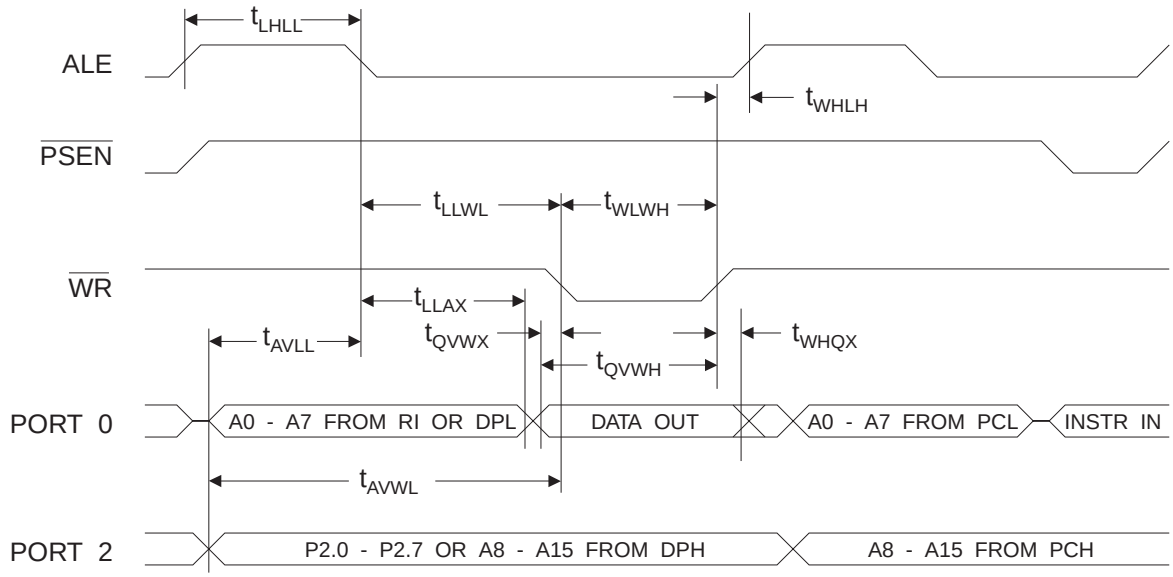
External Program Memory Read Cycle



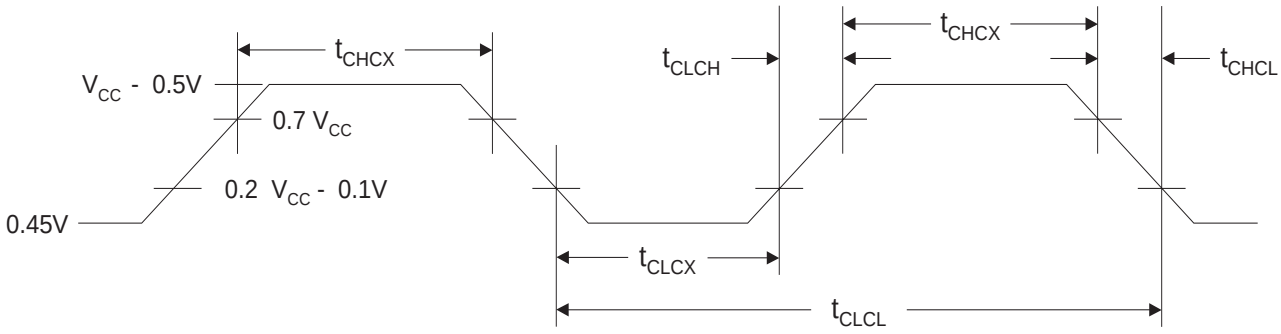
External Data Memory Read Cycle



External Data Memory Write Cycle



External Clock Drive Waveforms



External Clock Drive

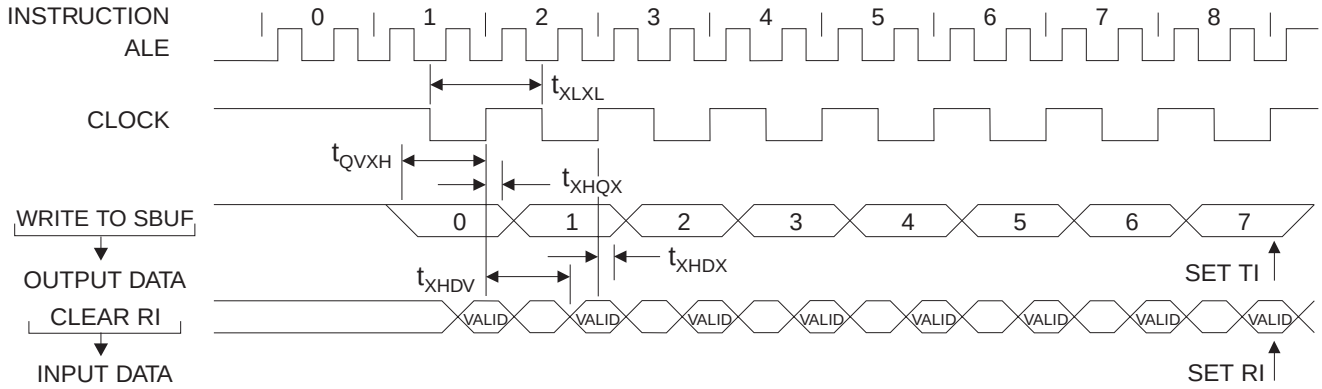
Symbol	Parameter	Min	Max	Units
$1/t_{CLCL}$	Oscillator Frequency	0	24	MHz
t_{CLCL}	Clock Period	41.6		ns
t_{CHCX}	High Time	15		ns
t_{CLCX}	Low Time	15		ns
t_{CLCH}	Rise Time		20	ns
t_{CHCL}	Fall Time		20	ns

Serial Port Timing: Shift Register Mode Test Conditions

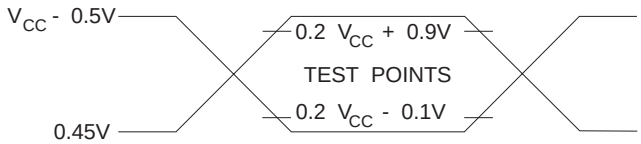
The values in this table are valid for V_{CC} = 5.0V ±20% and Load Capacitance = 80 pF.

Symbol	Parameter	12 MHz Osc		Variable Oscillator		Units
		Min	Max	Min	Max	
t_{XLXL}	Serial Port Clock Cycle Time	1.0		$12t_{CLCL}$		µs
t_{QVXH}	Output Data Setup to Clock Rising Edge	700		$10t_{CLCL}$-133		ns
t_{XHQX}	Output Data Hold After Clock Rising Edge	50		$2t_{CLCL}$-117		ns
t_{XHDX}	Input Data Hold After Clock Rising Edge	0		0		ns
t_{XHDV}	Clock Rising Edge to Input Data Valid		700		$10t_{CLCL}$-133	ns

Shift Register Mode Timing Waveforms

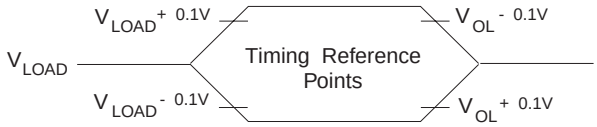


AC Testing Input/Output Waveforms(1)



Note: 1. AC Inputs during testing are driven at V_{CC} - 0.5V for a logic 1 and 0.45V for a logic 0. Timing measurements are made at V_{IH} min. for a logic 1 and V_{IL} max. for a logic 0.

Float Waveforms(1)



Note: 1. For timing purposes, a port pin is no longer floating when a 100 mV change from load voltage occurs. A port pin begins to float when a 100 mV change from the loaded V_{OH}/V_{OL} level occurs.

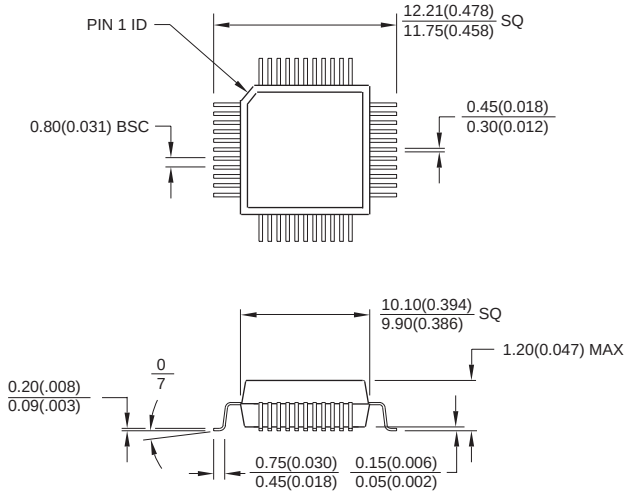
Ordering Information

Speed (MHz)	Power Supply	Ordering Code	Package	Operation Range
12	5V ±20%	AT89C52-12AC AT89C52-12JC AT89C52-12PC AT89C52-12QC	44A 44J 40P6 44Q	Commercial (0° C to 70° C)
		AT89C52-12AI AT89C52-12JI AT89C52-12PI AT89C52-12QI	44A 44J 40P6 44Q	Industrial (-40° C to 85° C)
16	5V ±20%	AT89C52-16AC AT89C52-16JC AT89C52-16PC AT89C52-16QC	44A 44J 40P6 44Q	Commercial (0° C to 70° C)
		AT89C52-16AI AT89C52-16JI AT89C52-16PI AT89C52-16QI	44A 44J 40P6 44Q	Industrial (-40° C to 85° C)
20	5V ±20%	AT89C52-20AC AT89C52-20JC AT89C52-20PC AT89C52-20QC	44A 44J 40P6 44Q	Commercial (0° C to 70° C)
		AT89C52-20AI AT89C52-20JI AT89C52-20PI AT89C52-20QI	44A 44J 40P6 44Q	Industrial (-40° C to 85° C)
24	5V ±20%	AT89C52-24AC AT89C52-24JC AT89C52-24PC AT89C52-24QC	44A 44J 40P6 44Q	Commercial (0° C to 70° C)
		AT89C52-24AI AT89C52-24JI AT89C52-24PI AT89C52-24QI	44A 44J 40P6 44Q	Industrial (-40° C to 85° C)

Package Type	
44A	44-lead, Thin Plastic Gull Wing Quad Flatpack (TQFP)
44J	44-lead, Plastic J-leaded Chip Carrier (PLCC)
40P6	40-lead, 0.600" Wide, Plastic Dual Inline Package (PDIP)
44Q	44-lead, Plastic Gull Wing Quad Flatpack (PQFP)

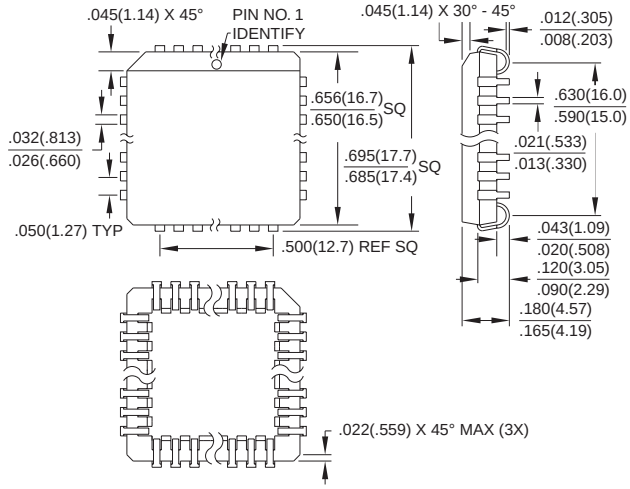
Packaging Information

44A, 44-lead, Thin (1.0 mm) Plastic Gull Wing Quad Flatpack (TQFP)
Dimensions in Millimeters and (Inches)*
JEDEC STANDARD MS-026 ACB

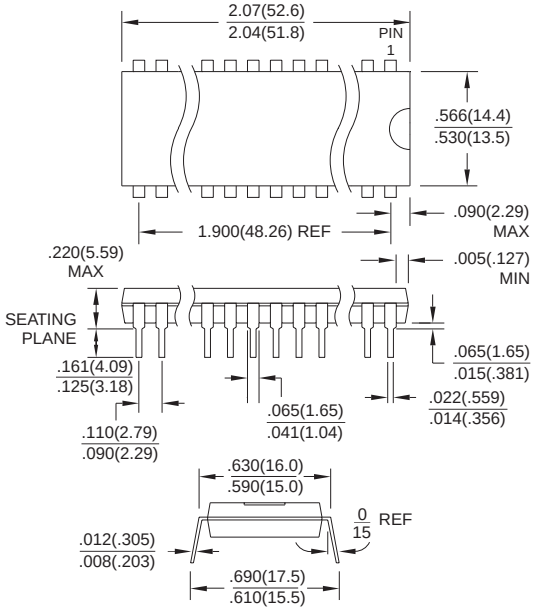


Controlling dimension: millimeters

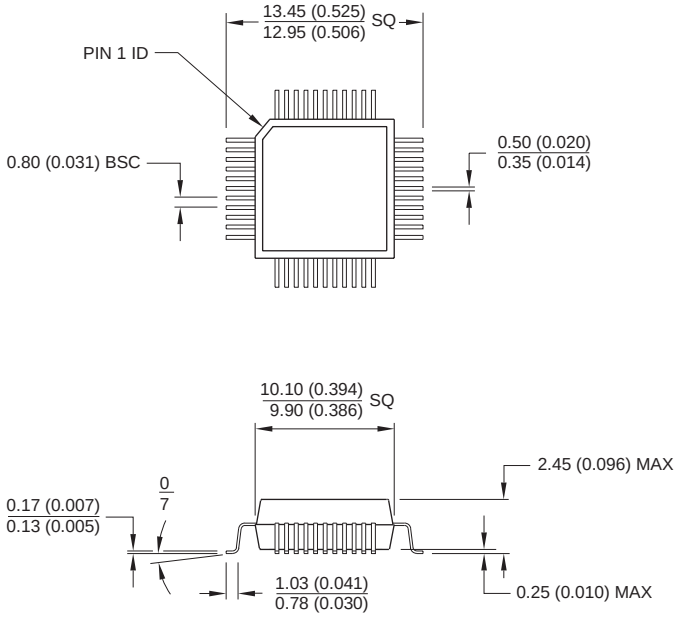
44J, 44-lead, Plastic J-leaded Chip Carrier (PLCC)
Dimensions in Inches and (Millimeters)
JEDEC STANDARD MS-018 AC



40P6, 40-lead, 0.600" Wide, Plastic Dual Inline Package (PDIP)
Dimensions in Inches and (Millimeters)



44Q, 44-lead, Plastic Quad Flat Package (PQFP)
Dimensions in Millimeters and (Inches)*
JEDEC STANDARD MS-022 AB



Controlling dimension: millimeters

Atmel Headquarters

Corporate Headquarters
2325 Orchard Parkway
San Jose, CA 95131
TEL (408) 441-0311
FAX (408) 487-2600

Europe
Atmel U.K., Ltd.
Coliseum Business Centre
Riverside Way
Camberley, Surrey GU15 3YL
England
TEL (44) 1276-686-677
FAX (44) 1276-686-697

Asia
Atmel Asia, Ltd.
Room 1219
Chinachem Golden Plaza
77 Mody Road Tsimhatsui
East Kowloon
Hong Kong
TEL (852) 2721-9778
FAX (852) 2722-1369

Japan
Atmel Japan K.K.
9F, Tonetsu Shinkawa Bldg.
1-24-8 Shinkawa
Chuo-ku, Tokyo 104-0033
Japan
TEL (81) 3-3523-3551
FAX (81) 3-3523-7581

Atmel Operations

Atmel Colorado Springs
1150 E. Cheyenne Mtn. Blvd.
Colorado Springs, CO 80906
TEL (719) 576-3300
FAX (719) 540-1759

Atmel Rousset
Zone Industrielle
13106 Rousset Cedex
France
TEL (33) 4-4253-6000
FAX (33) 4-4253-6001

Fax-on-Demand
North America:
1-(800) 292-8635
International:
1-(408) 441-0732

e-mail
literature@atmel.com

Web Site
http://www.atmel.com

BBS
1-(408) 436-4309

0313H–02/00/xM

FAIRCHILD SEMICONDUCTOR™

October 1987
Revised March 2002

CD4017BC • CD4022BC
Decade Counter/Divider with 10 Decoded Outputs • Divide-by-8 Counter/Divider with 8 Decoded Outputs

General Description

The CD4017BC is a 5-stage divide-by-10 Johnson counter with 10 decoded outputs and a carry out bit.

The CD4022BC is a 4-stage divide-by-8 Johnson counter with 8 decoded outputs and a carry-out bit.

These counters are cleared to their zero count by a logical "1" on their reset line. These counters are advanced on the positive edge of the clock signal when the clock enable signal is in the logical "0" state.

The configuration of the CD4017BC and CD4022BC permits medium speed operation and assures a hazard free counting sequence. The 10/8 decoded outputs are normally in the logical "0" state and go to the logical "1" state only at their respective time slot. Each decoded output remains high for 1 full clock cycle. The carry-out signal completes a full cycle for every 10/8 clock input cycles and is used as a ripple carry signal to any succeeding stages.

Features

- Wide supply voltage range: 3.0V to 15V
- High noise immunity: 0.45 V_{DD} (typ.)
- Low power Fan out of 2 driving 74L
  TTL compatibility: or 1 driving 74LS
- Medium speed operation: 5.0 MHz (typ.)
  with 10V V_{DD}
- Low power: 10 μW (typ.)
- Fully static operation

Applications

- Automotive
- Instrumentation
- Medical electronics
- Alarm systems
- Industrial electronics
- Remote metering

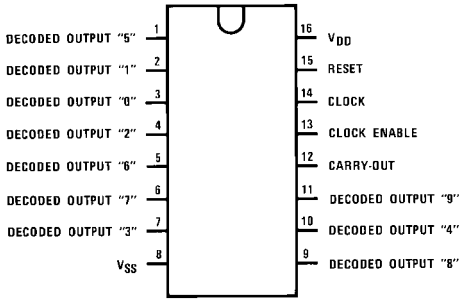
Ordering Code:

Order Number	Package Number	Package Description
CD4017BCM	M16A	16-Lead Small Outline Integrated Circuit (SOIC), JEDEC MS-012, 0.150" Narrow
CD4017BCSJ	M16D	16-Lead Small Outline Package (SOP), EIAJ TYPE II, 5.3mm Wide
CD4017BCN	N16E	16-Lead Plastic Dual-In-Line Package (PDIP), JEDEC MS-001, 0.300" Wide
CD4022BCM	M16A	16-Lead Small Outline Integrated Circuit (SOIC), JEDEC MS-012, 0.150" Narrow
CD4022BCN	N16E	16-Lead Plastic Dual-In-Line Package (PDIP), JEDEC MS-001, 0.300" Wide

Devices also available in Tape and Reel. Specify by appending the suffix letter "X" to the ordering code.

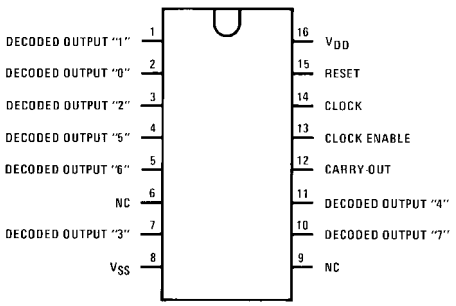
Connection Diagrams

Pin Assignments for DIP, SOIC and SOP
CD4017B



Top View

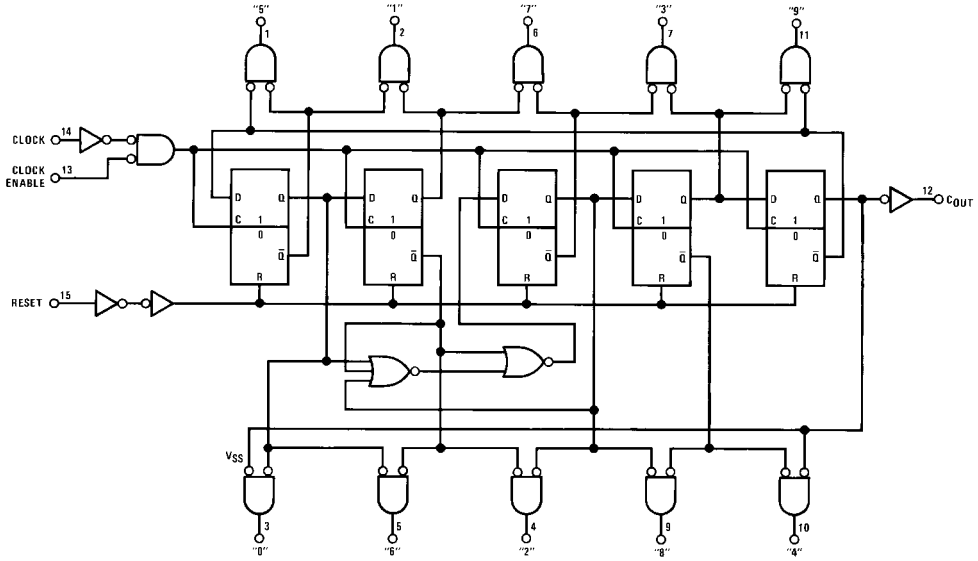
Pin Assignments for DIP and SOIC
CD4022B



Top View

Logic Diagrams

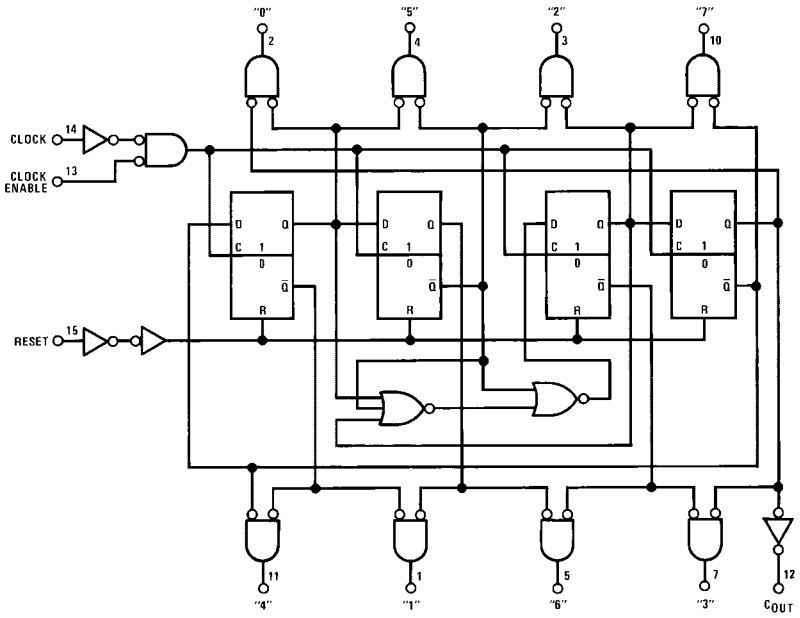
CD4017B



Terminal No. 8 = GND

Terminal No. 16 = V_{DD}

CD4022B



Terminal No. 16 = V_{DD}

Terminal No. 8 = GND

Absolute Maximum Ratings (Note 1)

(Note 2)

DC Supply Voltage (V_{DD})	−0.5 V_{DC} to +18 V_{DC}
Input Voltage (V_{IN})	−0.5 V_{DC} to V_{DD} +0.5 V_{DC}
Storage Temperature (T_S)	−65°C to +150°C
Power Dissipation (P_D)	
Dual-In-Line	700 mW
Small Outline	500 mW
Lead Temperature (T_L)	
(Soldering, 10 seconds)	260°C

Recommended Operating Conditions (Note 2)

DC Supply Voltage (V_{DD})	+3 V_{DC} to +15 V_{DC}
Input Voltage (V_{IN})	0 to V_{DD} V_{DC}
Operating Temperature Range (T_A)	−55°C to +125°C

Note 1: "Absolute Maximum Ratings" are those values beyond which the safety of the device cannot be guaranteed, they are not meant to imply that the devices should be operated at these limits. The table of "Recommended Operating Conditions" and "Electrical Characteristics" provides conditions for actual device operation.

Note 2: V_{SS} = 0V unless otherwise specified.

DC Electrical Characteristics (Note 2)

Symbol	Parameter	Conditions	−55°C		+25°			+125°C		Units
			Min	Max	Min	Typ	Max	Min	Max	
I_{DD}	Quiescent Device	V_{DD} = 5V		5		0.3	5		150	
	Current	V_{DD} = 10V		10		0.5	10		300	µA
		V_{DD} = 15V		20		1.0	20		600	
V_{OL}	LOW Level	$\|I_O\|$ < 1.0 µA								
	Output Voltage	V_{DD} = 5V		0.05		0	0.05		0.05	
		V_{DD} = 10V		0.05		0	0.05		0.05	V
		V_{DD} = 15V		0.05		0	0.05		0.05	
V_{OH}	HIGH Level	$\|I_O\|$ < 1.0 µA								
	Output Voltage	V_{DD} = 5V	4.95		4.95	5		4.95		
		V_{DD} = 10V	9.95		9.95	10		9.95		V
		V_{DD} = 15V	14.95		14.95	15		14.95		
V_{IL}	LOW Level	$\|I_O\|$ < 1.0 µA								
	Input Voltage	V_{DD} = 5V, V_O = 0.5V or 4.5V		1.5			1.5		1.5	
		V_{DD} = 10V, V_O = 1.0V or 9.0V		3.0			3.0		3.0	V
		V_{DD} = 15V, V_O = 1.5V or 13.5V		4.0			4.0		4.0	
V_{IH}	HIGH Level	$\|I_O\|$ < 1.0 µA								
	Input Voltage	V_{DD} = 5V, V_O = 0.5V or 4.5V	3.5		3.5			3.5		
		V_{DD} = 10V, V_O = 1.0V or 9.0V	7.0		7.0			7.0		V
		V_{DD} = 15V, V_O = 1.5V or 13.5V	11.0		11.0			11.0		
I_{OL}	LOW Level Output	V_{DD} = 5V, V_O = 0.4V	0.64		0.51	0.88		0.36		
	Current (Note 3)	V_{DD} = 10V, V_O = 0.5V	1.6		1.3	2.25		0.9		mA
		V_{DD} = 15V, V_O = 1.5V	4.2		3.4	8.8		2.4		
I_{OH}	HIGH Level Output	V_{DD} = 5V, V_O = 4.6V	−0.25		−0.2	−0.36		−0.14		
	Current (Note 3)	V_{DD} = 10V, V_O = 9.5V	−0.62		−0.5	−0.9		−0.35		mA
		V_{DD} = 15V, V_O = 13.5V	−1.8		−1.5	−3.5		−1.1		
I_{IN}	Input Current	V_{DD} = 15V, V_{IN} = 0V		−0.1		-10^{-5}	−0.1		−1.0	µA
		V_{DD} = 15V, V_{IN} = 15V		0.1		10^{-5}	0.1		1.0	

Note 3: I_{OL} and I_{OH} are tested one output at a time.

AC Electrical Characteristics (Note 4)

T_A= 25°C, C_L= 50 pF, R_L= 200k, t_{rCL} and t_{fCL}= 20 ns, unless otherwise specified

Symbol	Parameter	Conditions		Min	Typ	Max	Units
CLOCK OPERATION							
t_{PHL}, t_{PLH}	Propagation Delay Time Carry Out Line	V_{DD} = 5V			415	800	
		V_{DD} = 10V			160	320	ns
		V_{DD} = 15V			130	250	
	Carry Out Line	V_{DD} = 5V			240	480	
		V_{DD} = 10V	C_L = 15 pF		85	170	ns
		V_{DD} = 15V			70	140	
	Decode Out Lines	V_{DD} = 5V			500	1000	
		V_{DD} = 10V			200	400	ns
		V_{DD} = 15V			160	320	
t_{TLH}, t_{THL}	Transition Time Carry Out and Decode Out Lines						
	t_{TLH}	V_{DD} = 5V			200	360	
		V_{DD} = 10V			100	180	ns
		V_{DD} = 15V			80	130	
	t_{THL}	V_{DD} = 5V			100	200	
		V_{DD} = 10V			50	100	ns
		V_{DD} = 15V			40	80	
f_{CL}	Maximum Clock Frequency	V_{DD} = 5V	Measured with	1.0	2		
		V_{DD} = 10V	Respect to Carry	2.5	5		MHz
		V_{DD} = 15V	Output Line	3.0	6		
t_{WL}, t_{WH}	Minimum Clock Pulse Width	V_{DD} = 5V			125	250	
		V_{DD} = 10V			45	90	ns
		V_{DD} = 15V			35	70	
t_{rCL}, t_{fCL}	Clock Rise and Fall Time	V_{DD} = 5V				20	
		V_{DD} = 10V				15	µs
		V_{DD} = 15V				5	
t_{SU}	Minimum Clock Inhibit Data Setup Time	V_{DD} = 5V			120	240	
		V_{DD} = 10V			40	80	ns
		V_{DD} = 15V			32	65	
C_{IN}	Average Input Capacitance				5	7.5	pF

Note 4: AC Parameters are guaranteed by DC correlated testing.

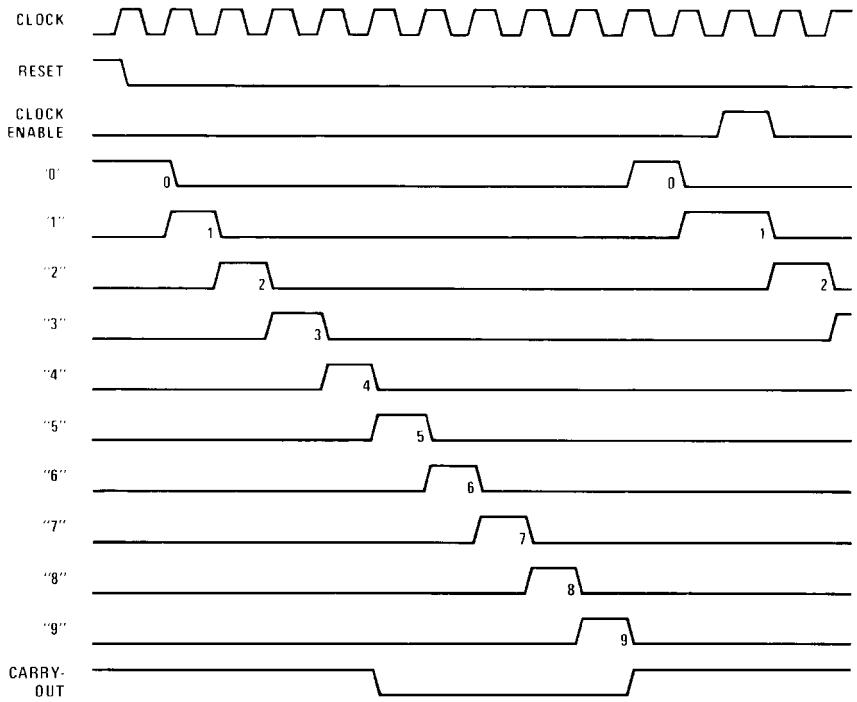
AC Electrical Characteristics (Note 4)

T_A = 25°C, C_L = 50 pF, R_L = 200k, t_{rCL} and t_{fCL} = 20 ns, unless otherwise specified

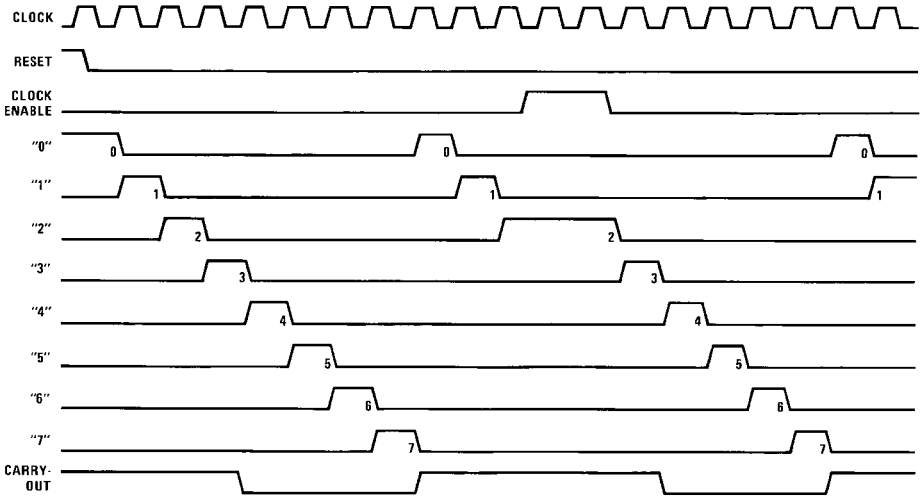
Symbol	Parameter	Conditions		Min	Typ	Max	Units
RESET OPERATION							
t_{PHL}, t_{PLH}	Propagation Delay Time						
	Carry Out Line	V_{DD} = 5V			415	800	
		V_{DD} = 10V			160	320	ns
		V_{DD} = 15V			130	250	
	Carry Out Line	V_{DD} = 5V			240	480	
		V_{DD} = 10V	C_L = 15 pF		85	170	ns
		V_{DD} = 15V			70	140	
	Decode Out Lines	V_{DD} = 5V			500	1000	
		V_{DD} = 10V			200	400	ns
		V_{DD} = 15V			160	320	
t_W	Minimum Reset	V_{DD} = 5V			200	400	
	Pulse Width	V_{DD} = 10V			70	140	ns
		V_{DD} = 15V			55	110	
t_{REM}	Minimum Reset	V_{DD} = 5V			75	150	
	Removal Time	V_{DD} = 10V			30	60	ns
		V_{DD} = 15V			25	50	

Timing Diagrams

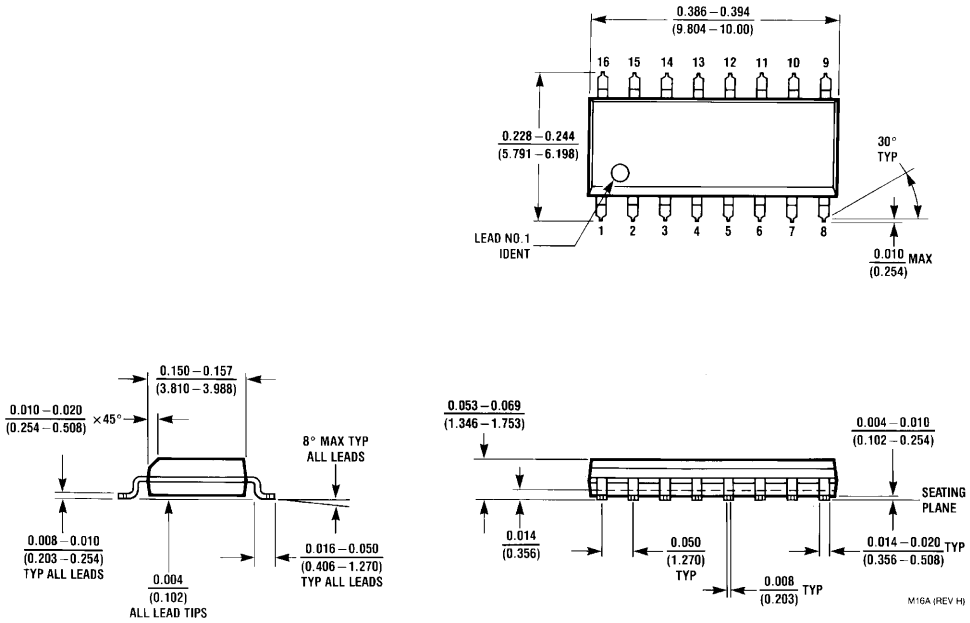
CD4017B



CD4022B

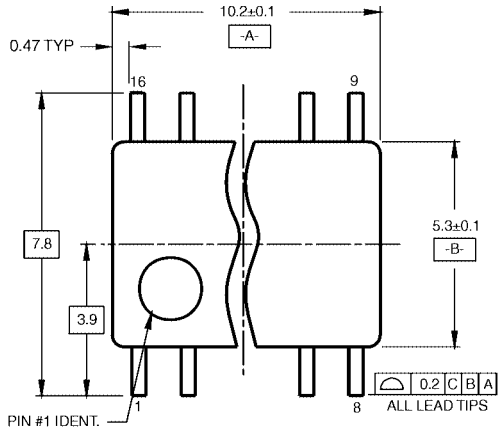


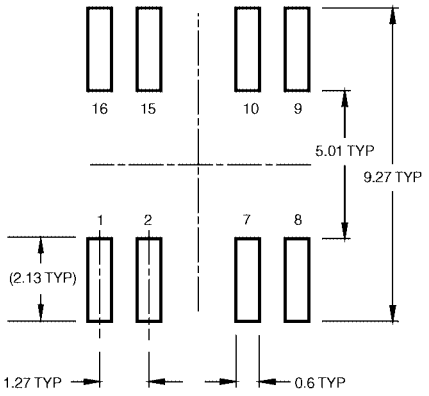
Physical Dimensions inches (millimeters) unless otherwise noted



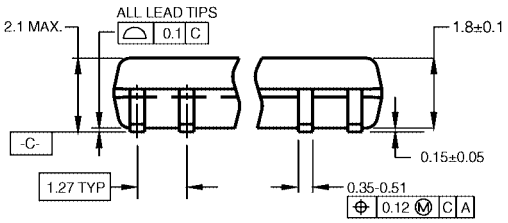
16-Lead Small Outline Integrated Circuit (SOIC), JEDEC MS-012, 0.150" Narrow
Package Number M16A

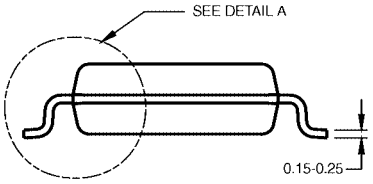
Physical Dimensions inches (millimeters) unless otherwise noted (Continued)





LAND PATTERN RECOMMENDATION





DIMENSIONS ARE IN MILLIMETERS

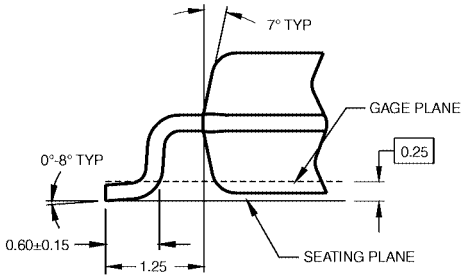
NOTES:

A. CONFORMS TO EIAJ EDR-7320 REGISTRATION, ESTABLISHED IN DECEMBER, 1998.

B. DIMENSIONS ARE IN MILLIMETERS.

C. DIMENSIONS ARE EXCLUSIVE OF BURRS, MOLD FLASH, AND TIE BAR EXTRUSIONS.

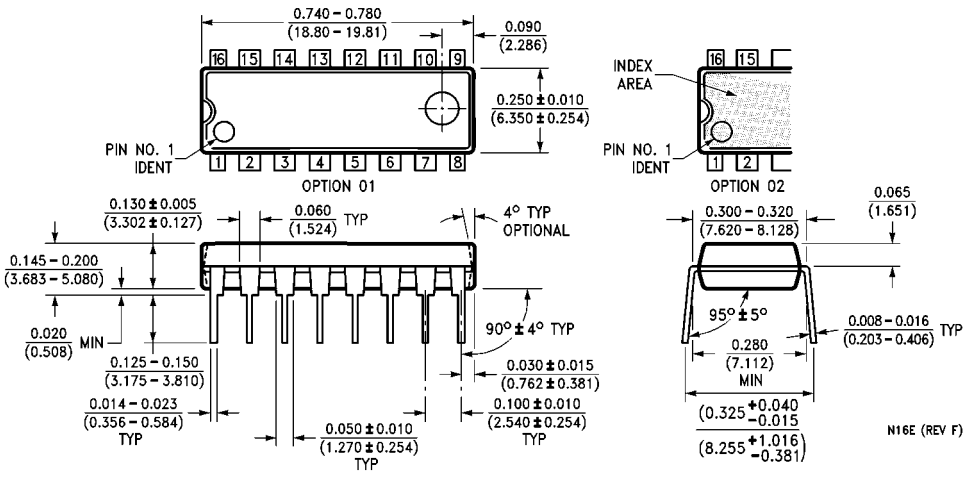
M16DRevB1



DETAIL A

16-Lead Small Outline Package (SOP), EIAJ TYPE II, 5.3mm Wide
Package Number M16D

Physical Dimensions inches (millimeters) unless otherwise noted (Continued)



16-Lead Plastic Dual-In-Line Package (PDIP), JEDEC MS-001, 0.300" Wide
Package Number N16E

DATA SHEET

For a complete data sheet, please also download:

- The IC04 LOCMOS HE4000B Logic Family Specifications HEF, HEC
- The IC04 LOCMOS HE4000B Logic Package Outlines/Information HEF, HEC

HEF4519B
MSI
Quadruple 2-input multiplexer

Product specification
File under Integrated Circuits, IC04

January 1995

Philips Semiconductors

PHILIPS

Quadruple 2-input multiplexer

HEF4519B
MSI

DESCRIPTION

The HEF4519B provides four multiplexing circuits with common select inputs (S_A, S_B); each circuit contains two inputs (A_n, B_n) and one output (O_n). It may be used to select four bits of information from one of two sources.

The 'A' inputs are selected when S_A is HIGH, the 'B' inputs when S_B is HIGH. When S_A and S_B are HIGH, the output (O_n) is the logical EXCLUSIVE-NOR of the A_n and B_n inputs ($O_n = A_n \odot B_n$).

When S_A and S_B are LOW, the output (O_n) is LOW, independent of the multiplexer inputs (A_n and B_n).

The HEF4519B cannot be used to multiplex analogue signals. The outputs utilize standard buffers for best performance.

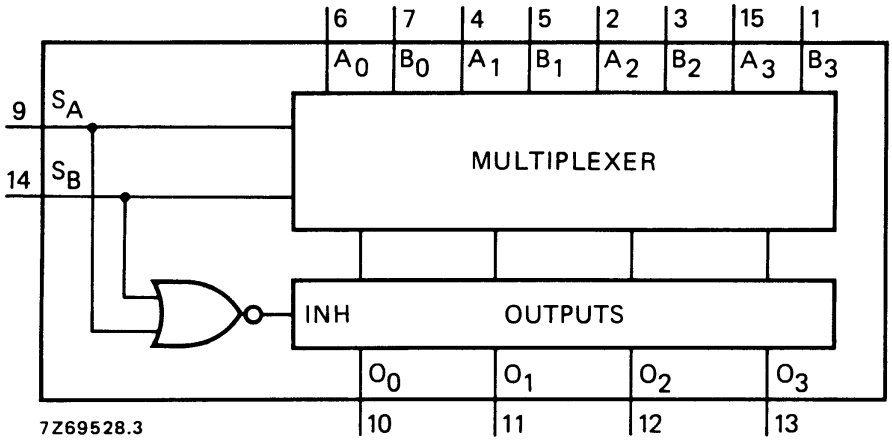


Fig.1 Functional diagram.

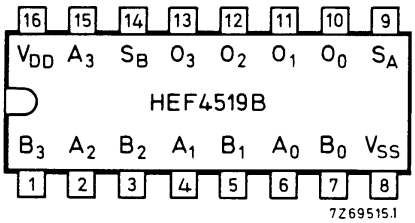


Fig.2 Pinning diagram.

PINNING

S_A, S_B	selects inputs (active HIGH)
A_0 to A_3	multiplexer inputs
B_0 to B_3	multiplexer inputs
O_0 to O_3	multiplexer outputs

FAMILY DATA, I_{DD} LIMITS category MSI

See Family Specifications

HEF4519BP(N): 16-lead DIL; plastic (SOT38-1)
HEF4519BD(F): 16-lead DIL; ceramic (cerdip) (SOT74)
HEF4519BT(D): 16-lead SO; plastic (SOT109-1)
( ): Package Designator North America

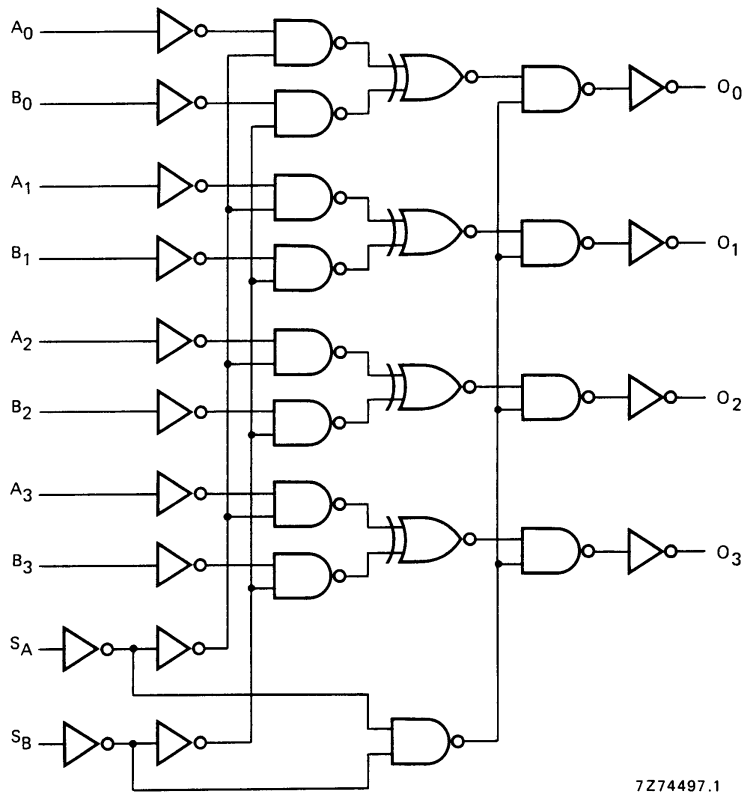


Fig.3 Logic diagram.

FUNCTION TABLE

INPUTS				OUTPUT
S_A	S_B	A_n	B_n	O_n
L	L	X	X	L
H	L	A_n	X	A_n
L	H	X	B_n	B_n
H	H	L	L	H
H	H	H	L	L
H	H	L	H	L
H	H	H	H	H

Notes

1. H = HIGH state (the more positive voltage)
   L = LOW state (the less positive voltage)
   X = state is immaterial

AC CHARACTERISTICS

V_{SS} = 0 V; T_{amb} = 25 °C; C_L = 50 pF; input transition times ≤ 20 ns

	V_{DD} V	SYMBOL	TYP.	MAX.		TYPICAL EXTRAPOLATION FORMULA
Propagation delays						
A_n , $B_n \rightarrow O_n$	5		95	190	ns	68 ns + (0,55 ns/pF) C_L
HIGH to LOW	10	t_{PHL}	40	80	ns	29 ns + (0,23 ns/pF) C_L
	15		30	60	ns	22 ns + (0,16 ns/pF) C_L
	5		80	160	ns	53 ns + (0,55 ns/pF) C_L
LOW to HIGH	10	t_{PLH}	40	80	ns	29 ns + (0,23 ns/pF) C_L
	15		30	60	ns	22 ns + (0,16 ns/pF) C_L
S_A, $S_B \rightarrow O_n$	5		95	190	ns	68 ns + (0,55 ns/pF) C_L
HIGH to LOW	10	t_{PHL}	40	80	ns	29 ns + (0,23 ns/pF) C_L
	15		30	55	ns	22 ns + (0,16 ns/pF) C_L
	5		85	165	ns	58 ns + (0,55 ns/pF) C_L
LOW to HIGH	10	t_{PLH}	40	80	ns	29 ns + (0,23 ns/pF) C_L
	15		30	60	ns	22 ns + (0,16 ns/pF) C_L
Output transition times	5		60	120	ns	10 ns + (1,0 ns/pF) C_L
HIGH to LOW	10	t_{THL}	30	60	ns	9 ns + (0,42 ns/pF) C_L
	15		20	40	ns	6 ns + (0,28 ns/pF) C_L
	5		60	120	ns	10 ns + (1,0 ns/pF) C_L
LOW to HIGH	10	t_{TLH}	30	60	ns	9 ns + (0,42 ns/pF) C_L
	15		20	40	ns	6 ns + (0,28 ns/pF) C_L

	V_{DD} V	TYPICAL FORMULA FOR P (μW)	
Dynamic power dissipation per package (P)	5	$1000\ f_i + \sum (f_o C_L) \times V_{DD}^2$	where f_i = input freq. (MHz) f_o = output freq. (MHz) C_L = load capacitance (pF) $\sum (f_o C_L)$ = sum of outputs V_{DD} = supply voltage (V)
	10	$6000\ f_i + \sum (f_o C_L) \times V_{DD}^2$	
	15	$17\ 000\ f_i + \sum (f_o C_L) \times V_{DD}^2$	

APPLICATION INFORMATION

Some examples of applications for the HEF4519B are:

- 2-input multiplexers.
- True/complement selectors.

Features

- Operating voltage: 2.4V~12V
- Low power and high noise immunity CMOS technology
- Low standby current
- Capable of decoding 12 bits of information
- Binary address setting
- Received codes are checked 3 times
- Address/Data number combination
  - HT12D: 8 address bits and 4 data bits
  - HT12F: 12 address bits only
- Built-in oscillator needs only 5% resistor
- Valid transmission indicator
- Easy interface with an RF or an infrared transmission medium
- Minimal external components
- Pair with Holtek's 2^{12} series of encoders
- 18-pin DIP, 20-pin SOP package

Applications

- Burglar alarm system
- Smoke and fire alarm system
- Garage door controllers
- Car door controllers
- Car alarm system
- Security system
- Cordless telephones
- Other remote control systems

General Description

The 2^{12} decoders are a series of CMOS LSIs for remote control system applications. They are paired with Holtek's 2^{12} series of encoders (refer to the encoder/decoder cross reference table). For proper operation, a pair of encoder/decoder with the same number of addresses and data format should be chosen.

The decoders receive serial addresses and data from a programmed 2^{12} series of encoders that are transmitted by a carrier using an RF or an IR transmission medium. They compare the serial input data three times continuously with their local addresses. If no error or unmatched codes are found, the input data codes are decoded and then transferred to the output pins. The VT pin also goes high to indicate a valid transmission.

The 2^{12} series of decoders are capable of decoding informations that consist of N bits of address and 12–N bits of data. Of this series, the HT12D is arranged to provide 8 address bits and 4 data bits, and HT12F is used to decode 12 bits of address information.

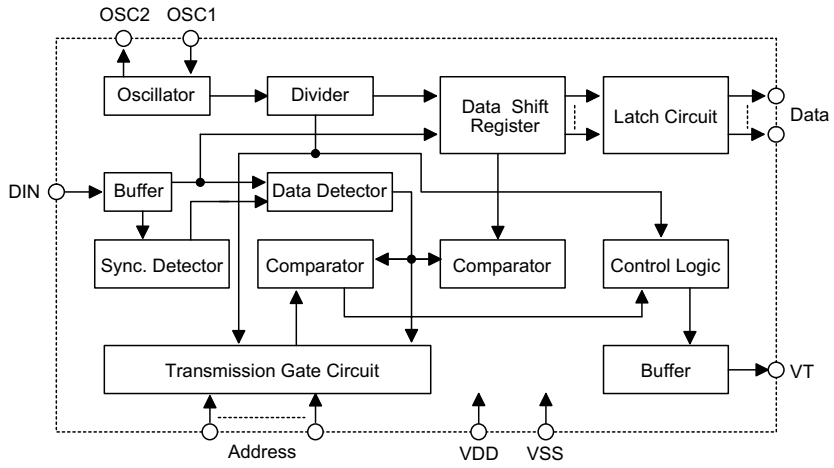
Selection Table

Function / Part No.	Address No.	Data		VT	Oscillator	Trigger	Package
		No.	Type				
HT12D	8	4	L	√	RC oscillator	DIN active ″Hi″	18DIP, 20SOP
HT12F	12	0	—	√	RC oscillator	DIN active ″Hi″	18DIP, 20SOP

Notes: Data type: L stands for latch type data output.

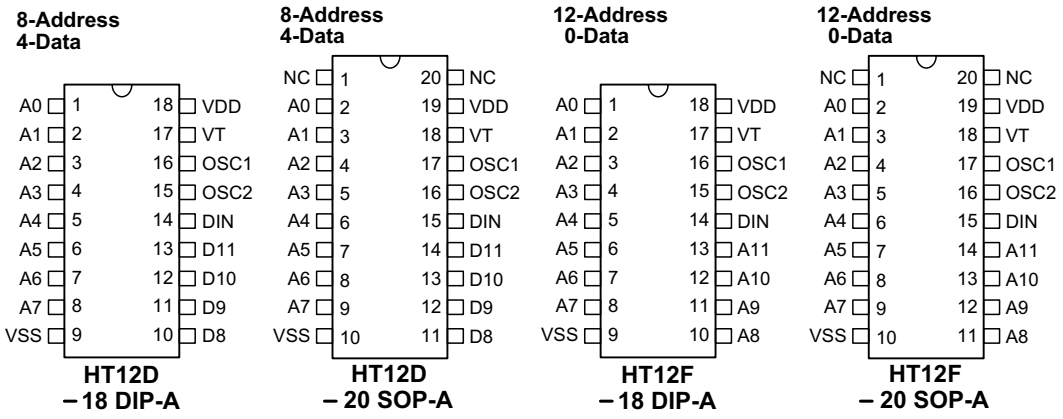
VT can be used as a momentary data output.

Block Diagram



Note: The address/data pins are available in various combinations (see the address/data table).

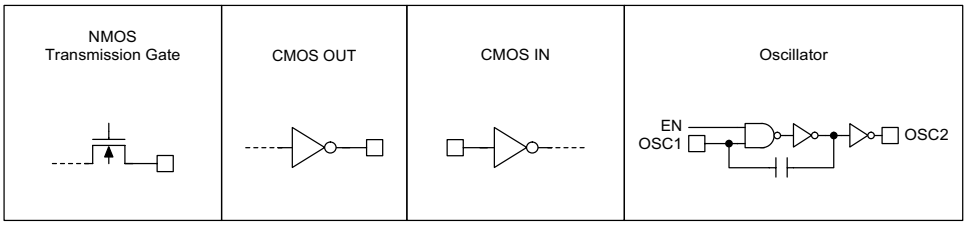
Pin Assignment



Pin Description

Pin Name	I/O	Internal Connection	Description
A0~A11 (HT12F)	I	NMOS Transmission Gate	Input pins for address A0~A11 setting These pins can be externally set to VSS or left open.
A0~A7 (HT12D)			Input pins for address A0~A7 setting These pins can be externally set to VSS or left open.
D8~D11 (HT12D)	O	CMOS OUT	Output data pins, power-on state is low.
DIN	I	CMOS IN	Serial data input pin
VT	O	CMOS OUT	Valid transmission, active high
OSC1	I	Oscillator	Oscillator input pin
OSC2	O	Oscillator	Oscillator output pin
VSS	—	—	Negative power supply, ground
VDD	—	—	Positive power supply

Approximate internal connection circuits



Absolute Maximum Ratings

Supply Voltage .....–0.3V to 13V

Input Voltage .....V_{SS}–0.3 to V_{DD}+0.3V

Storage Temperature .....–50°C to 125°C

Operating Temperature .....–20°C to 75°C

Note: These are stress ratings only. Stresses exceeding the range specified under ″Absolute Maximum Ratings″ may cause substantial damage to the device. Functional operation of this device at other conditions beyond those listed in the specification is not implied and prolonged exposure to extreme conditions may affect device reliability.

Electrical Characteristics

Ta=25°C

Symbol	Parameter	Test Conditions		Min.	Typ.	Max.	Unit
		V_{DD}	Conditions				
V_{DD}	Operating Voltage	—	—	2.4	5	12	V
I_{STB}	Standby Current	5V	Oscillator stops	—	0.1	1	μA
		12V		—	2	4	μA
I_{DD}	Operating Current	5V	No load, f_{OSC}=150kHz	—	200	400	μA
I_O	Data Output Source Current (D8~D11)	5V	V_{OH}=4.5V	–1	–1.6	—	mA
	Data Output Sink Current (D8~D11)	5V	V_{OL}=0.5V	1	1.6	—	mA
I_{VT}	VT Output Source Current	5V	V_{OH}=4.5V	–1	–1.6	—	mA
	VT Output Sink Current		V_{OL}=0.5V	1	1.6	—	mA
V_{IH}	″H″ Input Voltage	5V	—	3.5	—	5	V
V_{IL}	″L″ Input Voltage	5V	—	0	—	1	V
f_{OSC}	Oscillator Frequency	5V	R_{OSC}=51kΩ	—	150	—	kHz

Functional Description

Operation

The 2^{12} series of decoders provides various combinations of addresses and data pins in different packages so as to pair with the 2^{12} series of encoders.

The decoders receive data that are transmitted by an encoder and interpret the first N bits of code period as addresses and the last 12–N bits as data, where N is the address code number. A signal on the DIN pin activates the oscillator which in turn decodes the incoming address and data. The decoders will then check the received address three times continuously. If the received address codes all match the contents of the decoder′s local address, the 12–N bits of data are decoded to activate the output pins and the VT pin is set high to indicate a valid transmission. This will last unless the address code is incorrect or no signal is received.

The output of the VT pin is high only when the transmission is valid. Otherwise it is always low.

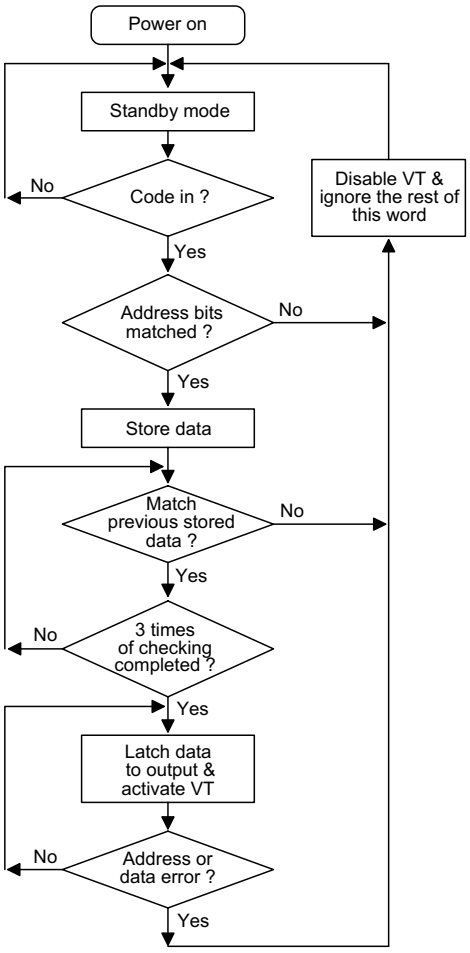
Output type

Of the 2^{12} series of decoders, the HT12F has no data output pin but its VT pin can be used as a momentary data output. The HT12D, on the other hand, provides 4 latch type data pins whose data remain unchanged until new data are received.

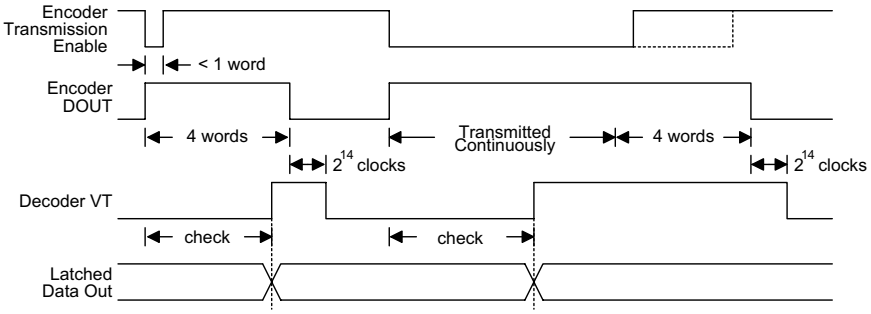
Part No.	Data Pins	Address Pins	Output Type	Operating Voltage
HT12D	4	8	Latch	2.4V~12V
HT12F	0	12	—	2.4V~12V

Flowchart

The oscillator is disabled in the standby state and activated when a logic ″high″ signal applies to the DIN pin. That is to say, the DIN should be kept low if there is no signal input.



Decoder timing



Encoder/Decoder cross reference table

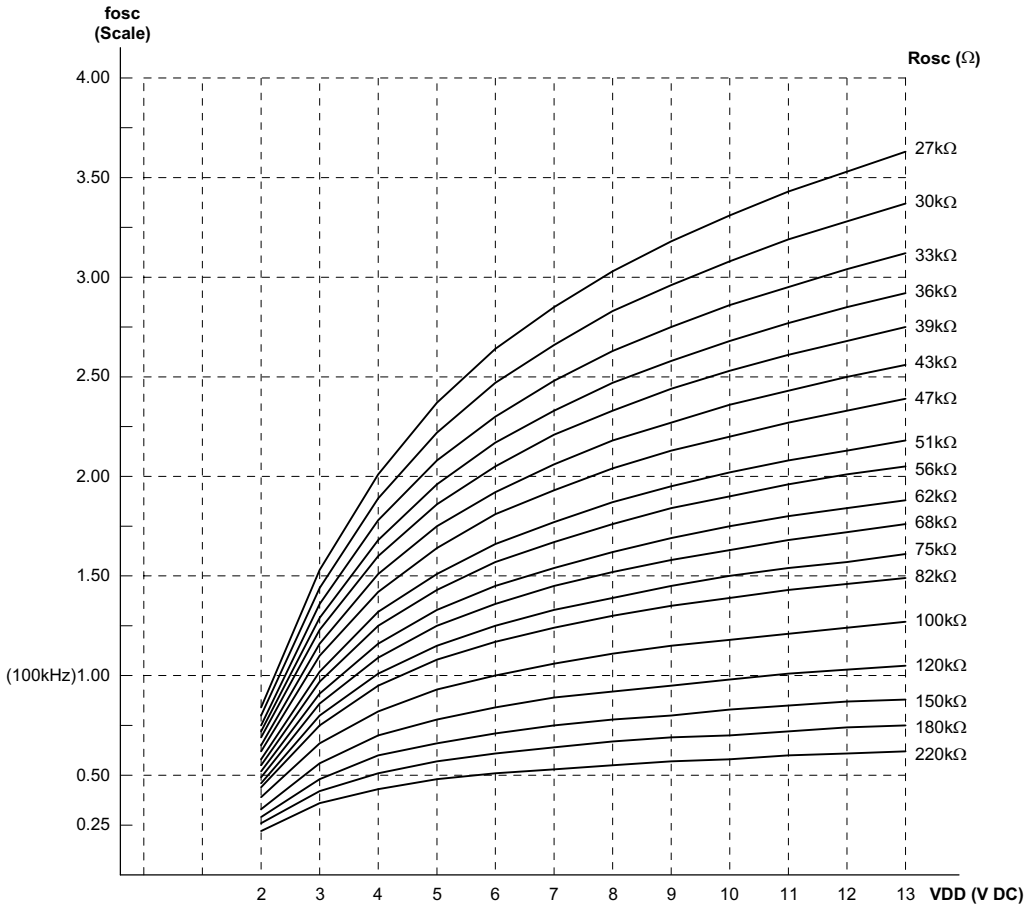
Decoders Part No.	Data Pins	Address Pins	VT	Pair Encoder	Package			
					Encoder		Decoder	
					DIP	SOP	DIP	SOP
HT12D	4	8	√	HT12A HT12E	18	20	18	20
HT12F	0	12	√	HT12A HT12E	18	20	18	20

Address/Data sequence

The following table provides address/data sequence for various models of the 2^{12} series of decoders.

Part No.	Address/Data Bits											
	0	1	2	3	4	5	6	7	8	9	10	11
HT12D	A0	A1	A2	A3	A4	A5	A6	A7	D8	D9	D10	D11
HT12F	A0	A1	A2	A3	A4	A5	A6	A7	A8	A9	A10	A11

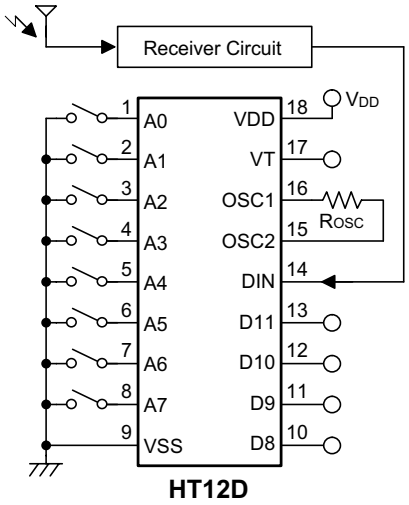
Oscillator frequency vs supply voltage



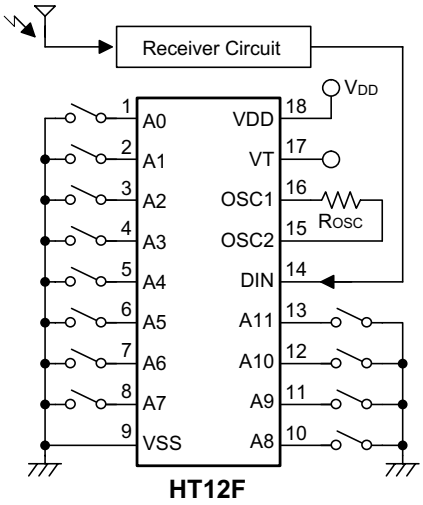
Note: The recommended oscillator frequency is f_{OSCD} (decoder) ≅ 50 f_{OSCE} (HT12E encoder)

≅ $\frac{1}{3}$ f_{OSCE} (HT12A encoder).

Application Circuits



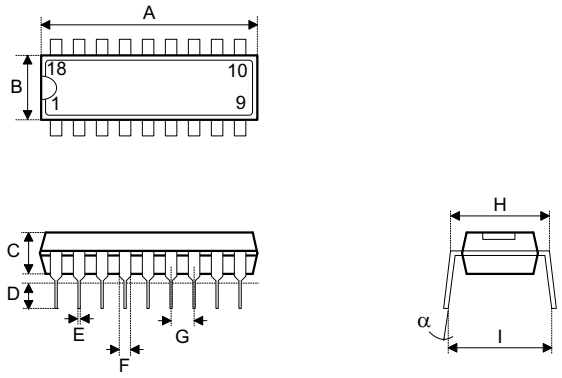
HT12D



HT12F

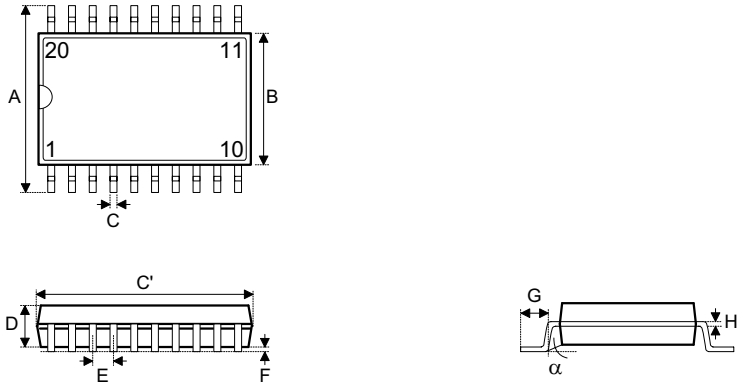
Package Information

18-pin DIP (300mil) outline dimensions



Symbol	Dimensions in mil		
	Min.	Nom.	Max.
A	895	—	915
B	240	—	260
C	125	—	135
D	125	—	145
E	16	—	20
F	50	—	70
G	—	100	—
H	295	—	315
I	335	—	375
α	0°	—	15°

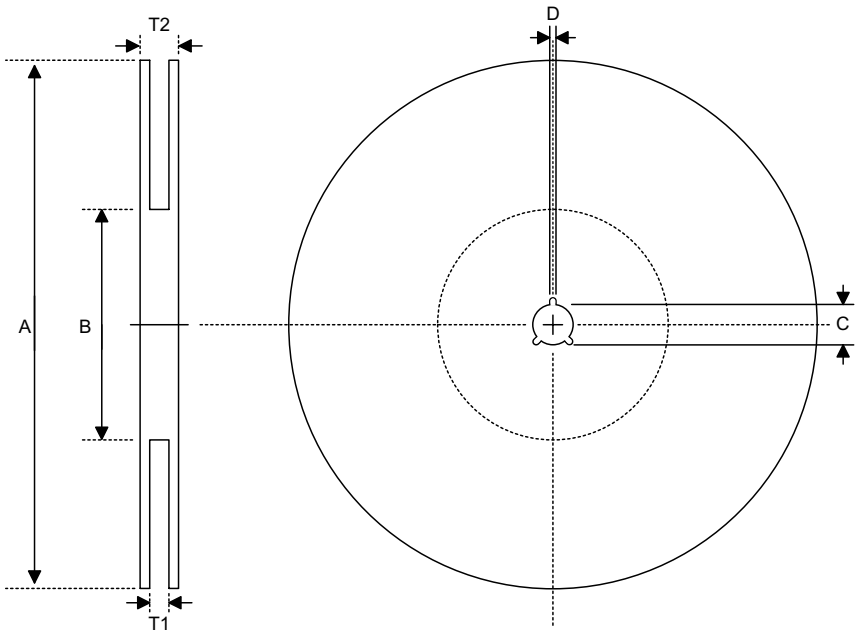
20-pin SOP (300mil) outline dimensions



Symbol	Dimensions in mil		
	Min.	Nom.	Max.
A	394	—	419
B	290	—	300
C	14	—	20
C′	490	—	510
D	92	—	104
E	—	50	—
F	4	—	—
G	32	—	38
H	4	—	12
α	0°	—	10°

Product Tape and Reel Specifications

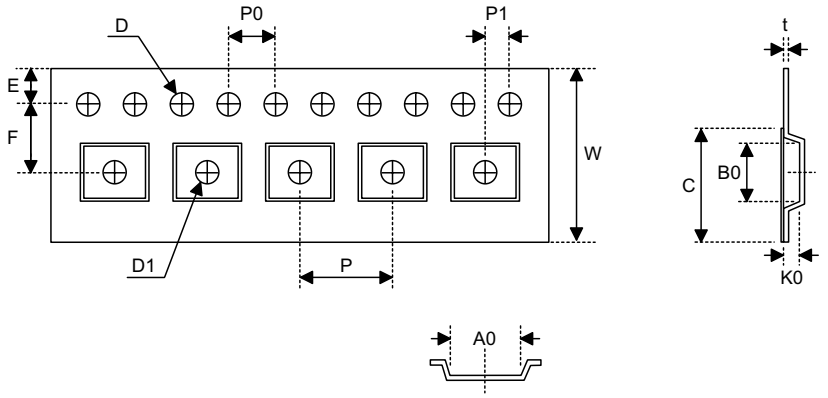
Reel dimensions



SOP 20W

Symbol	Description	Dimensions in mm
A	Reel Outer Diameter	330±1.0
B	Reel Inner Diameter	62±1.5
C	Spindle Hole Diameter	13.0+0.5 −0.2
D	Key Slit Width	2.0±0.5
T1	Space Between Flange	24.8+0.3 −0.2
T2	Reel Thickness	30.2±0.2

Carrier tape dimensions



SOP 20W

Symbol	Description	Dimensions in mm
W	Carrier Tape Width	24.0+0.3 −0.1
P	Cavity Pitch	12.0±0.1
E	Perforation Position	1.75±0.1
F	Cavity to Perforation (Width Direction)	11.5±0.1
D	Perforation Diameter	1.5+0.1
D1	Cavity Hole Diameter	1.5+0.25
P0	Perforation Pitch	4.0±0.1
P1	Cavity to Perforation (Length Direction)	2.0±0.1
A0	Cavity Length	10.8±0.1
B0	Cavity Width	13.3±0.1
K0	Cavity Depth	3.2±0.1
t	Carrier Tape Thickness	0.3±0.05
C	Cover Tape Width	21.3

Holtek Semiconductor Inc. (Headquarters)
No.3, Creation Rd. II, Science Park, Hsinchu, Taiwan
Tel: 886-3-563-1999
Fax: 886-3-563-1189
http://www.holtek.com.tw

Holtek Semiconductor Inc. (Taipei Sales Office)
4F-2, No. 3-2, YuanQu St., Nankang Software Park, Taipei 115, Taiwan
Tel: 886-2-2655-7070
Fax: 886-2-2655-7373
Fax: 886-2-2655-7383 (International sales hotline)

Holtek Semiconductor Inc. (Shanghai Sales Office)
7th Floor, Building 2, No.889, Yi Shan Rd., Shanghai, China 200233
Tel: 021-6485-5560
Fax: 021-6485-0313
http://www.holtek.com.cn

Holtek Semiconductor Inc. (Shenzhen Sales Office)
5/F, Unit A, Productivity Building, Cross of Science M 3rd Road and Gaoxin M 2nd Road, Science Park, Nanshan District, Shenzhen, China 518057
Tel: 0755-8616-9908, 8616-9308
Fax: 0755-8616-9533

Holtek Semiconductor Inc. (Beijing Sales Office)
Suite 1721, Jinyu Tower, A129 West Xuan Wu Men Street, Xicheng District, Beijing, China 100031
Tel: 010-6641-0030, 6641-7751, 6641-7752
Fax: 010-6641-0125

Holtek Semiconductor Inc. (Chengdu Sales Office)
709, Building 3, Champagne Plaza, No.97 Dongda Street, Chengdu, Sichuan, China 610016
Tel: 028-6653-6590
Fax: 028-6653-6591

Holmate Semiconductor, Inc. (North America Sales Office)
46729 Fremont Blvd., Fremont, CA 94538
Tel: 510-252-9880
Fax: 510-252-9885
http://www.holmate.com

Features

- Operating voltage: 2.4V~12V
- Low power and high noise immunity CMOS technology
- Low standby current
- Capable of decoding 12 bits of information
- Binary address setting
- Received codes are checked 3 times
- Address/Data number combination
  - HT12D: 8 address bits and 4 data bits
  - HT12F: 12 address bits only
- Built-in oscillator needs only 5% resistor
- Valid transmission indicator
- Easy interface with an RF or an infrared transmission medium
- Minimal external components
- Pair with Holtek's 2^{12} series of encoders
- 18-pin DIP, 20-pin SOP package

Applications

- Burglar alarm system
- Smoke and fire alarm system
- Garage door controllers
- Car door controllers
- Car alarm system
- Security system
- Cordless telephones
- Other remote control systems

General Description

The 2^{12} decoders are a series of CMOS LSIs for remote control system applications. They are paired with Holtek's 2^{12} series of encoders (refer to the encoder/decoder cross reference table). For proper operation, a pair of encoder/decoder with the same number of addresses and data format should be chosen.

The decoders receive serial addresses and data from a programmed 2^{12} series of encoders that are transmitted by a carrier using an RF or an IR transmission medium. They compare the serial input data three times continuously with their local addresses. If no error or unmatched codes are found, the input data codes are decoded and then transferred to the output pins. The VT pin also goes high to indicate a valid transmission.

The 2^{12} series of decoders are capable of decoding informations that consist of N bits of address and 12–N bits of data. Of this series, the HT12D is arranged to provide 8 address bits and 4 data bits, and HT12F is used to decode 12 bits of address information.

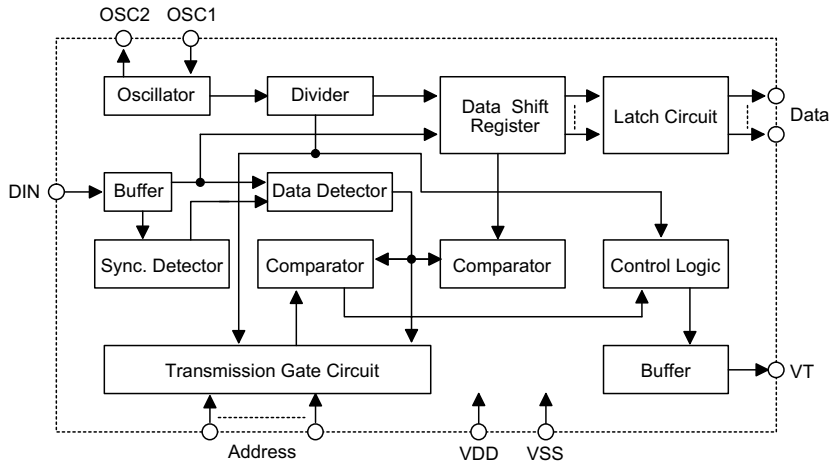
Selection Table

Function / Part No.	Address No.	Data		VT	Oscillator	Trigger	Package
		No.	Type				
HT12D	8	4	L	√	RC oscillator	DIN active ″Hi″	18DIP, 20SOP
HT12F	12	0	—	√	RC oscillator	DIN active ″Hi″	18DIP, 20SOP

Notes: Data type: L stands for latch type data output.

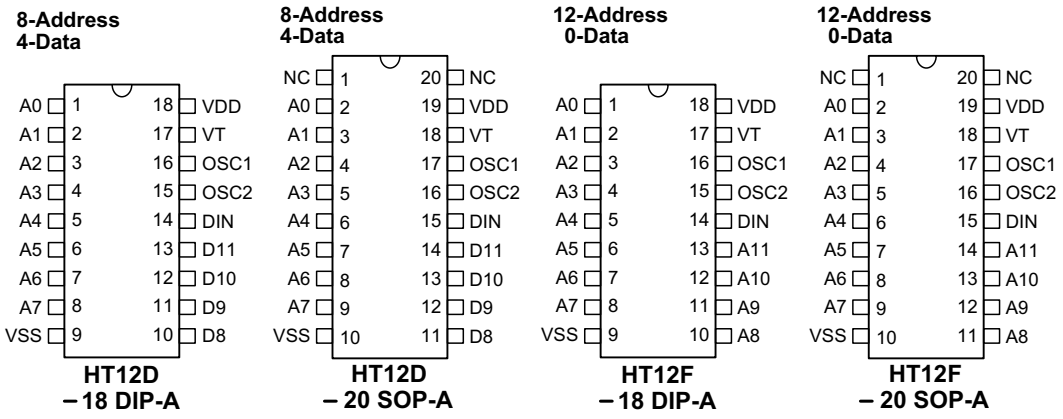
VT can be used as a momentary data output.

Block Diagram



Note: The address/data pins are available in various combinations (see the address/data table).

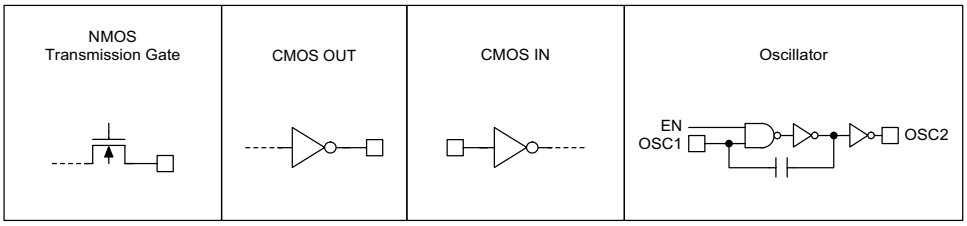
Pin Assignment



Pin Description

Pin Name	I/O	Internal Connection	Description
A0~A11 (HT12F)	I	NMOS Transmission Gate	Input pins for address A0~A11 setting These pins can be externally set to VSS or left open.
A0~A7 (HT12D)			Input pins for address A0~A7 setting These pins can be externally set to VSS or left open.
D8~D11 (HT12D)	O	CMOS OUT	Output data pins, power-on state is low.
DIN	I	CMOS IN	Serial data input pin
VT	O	CMOS OUT	Valid transmission, active high
OSC1	I	Oscillator	Oscillator input pin
OSC2	O	Oscillator	Oscillator output pin
VSS	—	—	Negative power supply, ground
VDD	—	—	Positive power supply

Approximate internal connection circuits



Absolute Maximum Ratings

Supply Voltage .....–0.3V to 13V

Input Voltage .....V_{SS}–0.3 to V_{DD}+0.3V

Storage Temperature .....–50°C to 125°C

Operating Temperature .....–20°C to 75°C

Note: These are stress ratings only. Stresses exceeding the range specified under ″Absolute Maximum Ratings″ may cause substantial damage to the device. Functional operation of this device at other conditions beyond those listed in the specification is not implied and prolonged exposure to extreme conditions may affect device reliability.

Electrical Characteristics

Ta=25°C

Symbol	Parameter	Test Conditions		Min.	Typ.	Max.	Unit
		V_{DD}	Conditions				
V_{DD}	Operating Voltage	—	—	2.4	5	12	V
I_{STB}	Standby Current	5V	Oscillator stops	—	0.1	1	μA
		12V		—	2	4	μA
I_{DD}	Operating Current	5V	No load, f_{OSC}=150kHz	—	200	400	μA
I_O	Data Output Source Current (D8~D11)	5V	V_{OH}=4.5V	–1	–1.6	—	mA
	Data Output Sink Current (D8~D11)	5V	V_{OL}=0.5V	1	1.6	—	mA
I_{VT}	VT Output Source Current	5V	V_{OH}=4.5V	–1	–1.6	—	mA
	VT Output Sink Current		V_{OL}=0.5V	1	1.6	—	mA
V_{IH}	″H″ Input Voltage	5V	—	3.5	—	5	V
V_{IL}	″L″ Input Voltage	5V	—	0	—	1	V
f_{OSC}	Oscillator Frequency	5V	R_{OSC}=51kΩ	—	150	—	kHz

Functional Description

Operation

The 2^{12} series of decoders provides various combinations of addresses and data pins in different packages so as to pair with the 2^{12} series of encoders.

The decoders receive data that are transmitted by an encoder and interpret the first N bits of code period as addresses and the last 12–N bits as data, where N is the address code number. A signal on the DIN pin activates the oscillator which in turn decodes the incoming address and data. The decoders will then check the received address three times continuously. If the received address codes all match the contents of the decoder′s local address, the 12–N bits of data are decoded to activate the output pins and the VT pin is set high to indicate a valid transmission. This will last unless the address code is incorrect or no signal is received.

The output of the VT pin is high only when the transmission is valid. Otherwise it is always low.

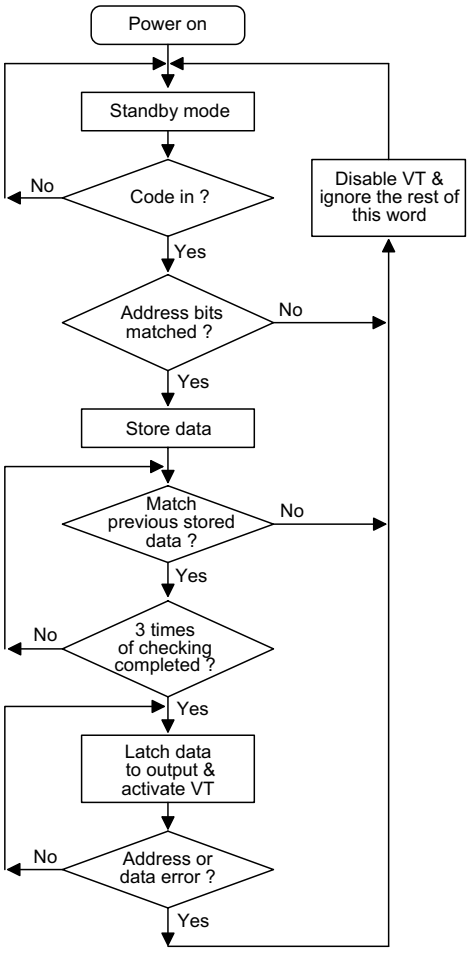
Output type

Of the 2^{12} series of decoders, the HT12F has no data output pin but its VT pin can be used as a momentary data output. The HT12D, on the other hand, provides 4 latch type data pins whose data remain unchanged until new data are received.

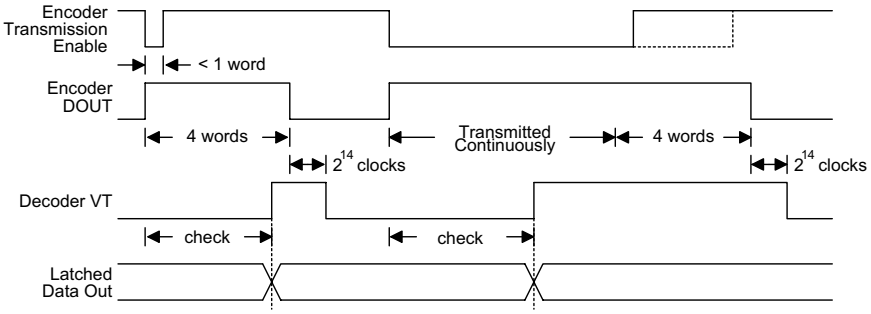
Part No.	Data Pins	Address Pins	Output Type	Operating Voltage
HT12D	4	8	Latch	2.4V~12V
HT12F	0	12	—	2.4V~12V

Flowchart

The oscillator is disabled in the standby state and activated when a logic ″high″ signal applies to the DIN pin. That is to say, the DIN should be kept low if there is no signal input.



Decoder timing



Encoder/Decoder cross reference table

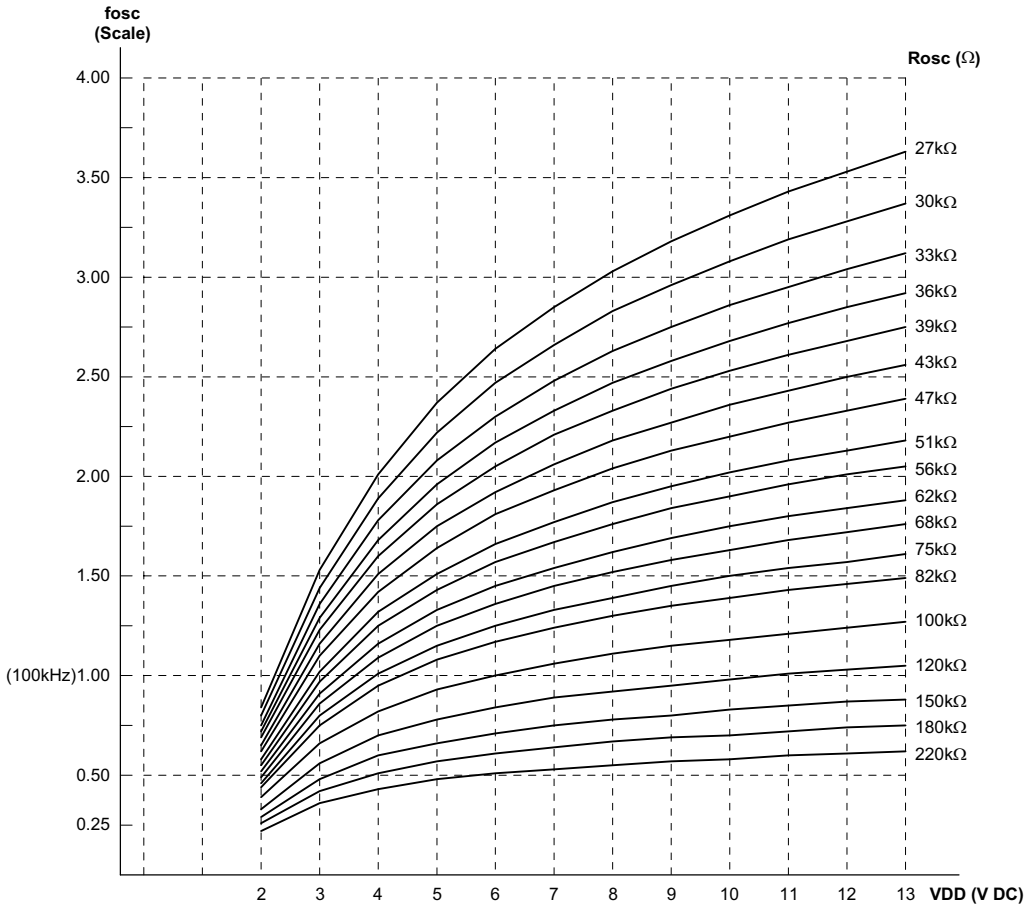
Decoders Part No.	Data Pins	Address Pins	VT	Pair Encoder	Package			
					Encoder		Decoder	
					DIP	SOP	DIP	SOP
HT12D	4	8	√	HT12A HT12E	18	20	18	20
HT12F	0	12	√	HT12A HT12E	18	20	18	20

Address/Data sequence

The following table provides address/data sequence for various models of the 2^{12} series of decoders.

Part No.	Address/Data Bits											
	0	1	2	3	4	5	6	7	8	9	10	11
HT12D	A0	A1	A2	A3	A4	A5	A6	A7	D8	D9	D10	D11
HT12F	A0	A1	A2	A3	A4	A5	A6	A7	A8	A9	A10	A11

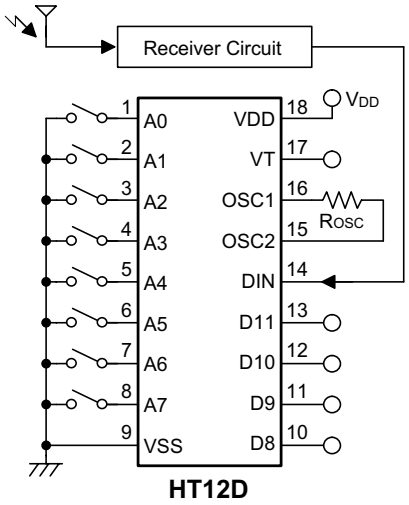
Oscillator frequency vs supply voltage



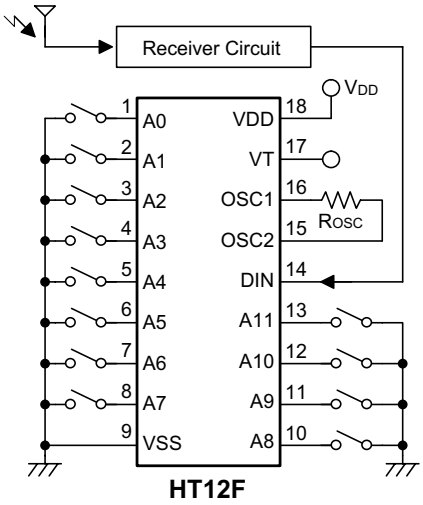
Note: The recommended oscillator frequency is f_{OSCD} (decoder) ≅ 50 f_{OSCE} (HT12E encoder)

≅ $\frac{1}{3}$ f_{OSCE} (HT12A encoder).

Application Circuits



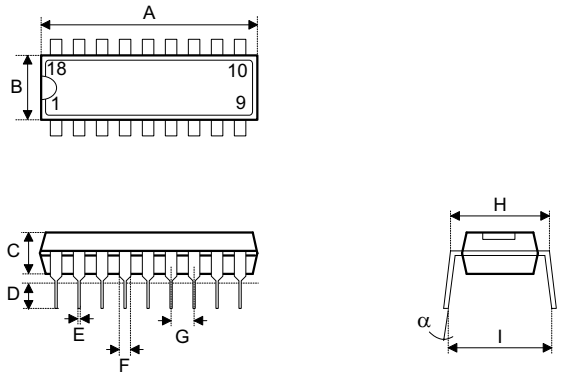
HT12D



HT12F

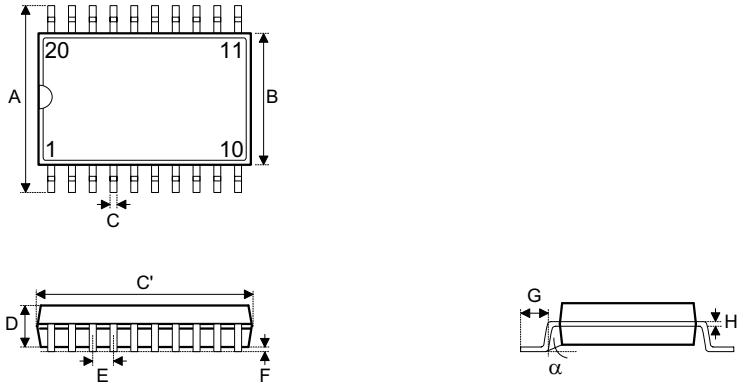
Package Information

18-pin DIP (300mil) outline dimensions



Symbol	Dimensions in mil		
	Min.	Nom.	Max.
A	895	—	915
B	240	—	260
C	125	—	135
D	125	—	145
E	16	—	20
F	50	—	70
G	—	100	—
H	295	—	315
I	335	—	375
α	0°	—	15°

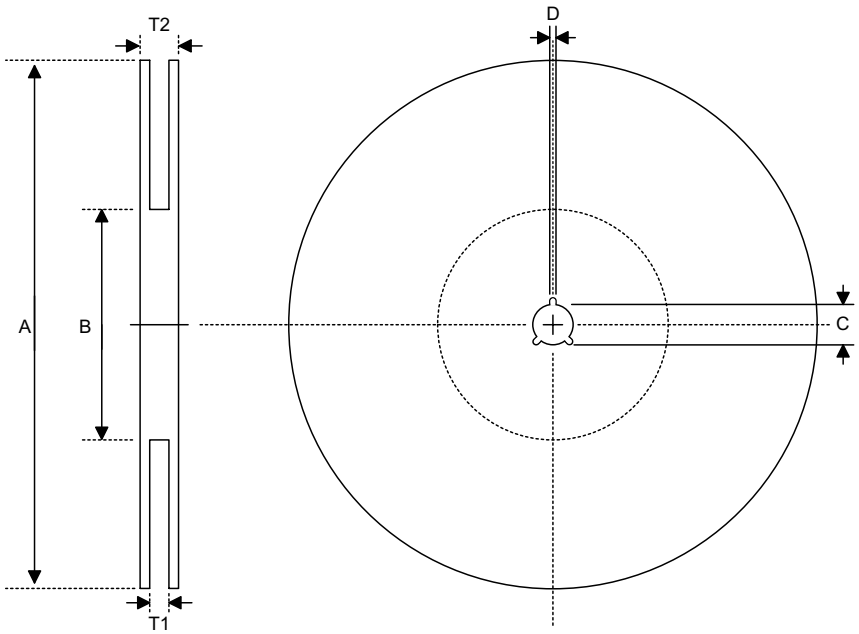
20-pin SOP (300mil) outline dimensions



Symbol	Dimensions in mil		
	Min.	Nom.	Max.
A	394	—	419
B	290	—	300
C	14	—	20
C′	490	—	510
D	92	—	104
E	—	50	—
F	4	—	—
G	32	—	38
H	4	—	12
α	0°	—	10°

Product Tape and Reel Specifications

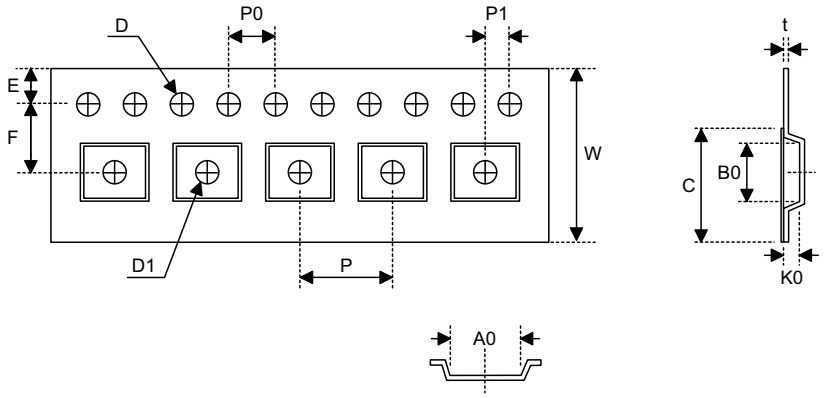
Reel dimensions



SOP 20W

Symbol	Description	Dimensions in mm
A	Reel Outer Diameter	330±1.0
B	Reel Inner Diameter	62±1.5
C	Spindle Hole Diameter	13.0+0.5 −0.2
D	Key Slit Width	2.0±0.5
T1	Space Between Flange	24.8+0.3 −0.2
T2	Reel Thickness	30.2±0.2

Carrier tape dimensions



SOP 20W

Symbol	Description	Dimensions in mm
W	Carrier Tape Width	24.0+0.3 −0.1
P	Cavity Pitch	12.0±0.1
E	Perforation Position	1.75±0.1
F	Cavity to Perforation (Width Direction)	11.5±0.1
D	Perforation Diameter	1.5+0.1
D1	Cavity Hole Diameter	1.5+0.25
P0	Perforation Pitch	4.0±0.1
P1	Cavity to Perforation (Length Direction)	2.0±0.1
A0	Cavity Length	10.8±0.1
B0	Cavity Width	13.3±0.1
K0	Cavity Depth	3.2±0.1
t	Carrier Tape Thickness	0.3±0.05
C	Cover Tape Width	21.3

Holtek Semiconductor Inc. (Headquarters)
No.3, Creation Rd. II, Science Park, Hsinchu, Taiwan
Tel: 886-3-563-1999
Fax: 886-3-563-1189
http://www.holtek.com.tw

Holtek Semiconductor Inc. (Taipei Sales Office)
4F-2, No. 3-2, YuanQu St., Nankang Software Park, Taipei 115, Taiwan
Tel: 886-2-2655-7070
Fax: 886-2-2655-7373
Fax: 886-2-2655-7383 (International sales hotline)

Holtek Semiconductor Inc. (Shanghai Sales Office)
7th Floor, Building 2, No.889, Yi Shan Rd., Shanghai, China 200233
Tel: 021-6485-5560
Fax: 021-6485-0313
http://www.holtek.com.cn

Holtek Semiconductor Inc. (Shenzhen Sales Office)
5/F, Unit A, Productivity Building, Cross of Science M 3rd Road and Gaoxin M 2nd Road, Science Park, Nanshan District, Shenzhen, China 518057
Tel: 0755-8616-9908, 8616-9308
Fax: 0755-8616-9533

Holtek Semiconductor Inc. (Beijing Sales Office)
Suite 1721, Jinyu Tower, A129 West Xuan Wu Men Street, Xicheng District, Beijing, China 100031
Tel: 010-6641-0030, 6641-7751, 6641-7752
Fax: 010-6641-0125

Holtek Semiconductor Inc. (Chengdu Sales Office)
709, Building 3, Champagne Plaza, No.97 Dongda Street, Chengdu, Sichuan, China 610016
Tel: 028-6653-6590
Fax: 028-6653-6591

Holmate Semiconductor, Inc. (North America Sales Office)
46729 Fremont Blvd., Fremont, CA 94538
Tel: 510-252-9880
Fax: 510-252-9885
http://www.holmate.com

HT12A/HT12E
2^{12} Series of Encoders

Features

- Operating voltage
  - 2.4V~5V for the HT12A
  - 2.4V~12V for the HT12E
- Low power and high noise immunity CMOS technology
- Low standby current: 0.1μA (typ.) at V_{DD}=5V
- HT12A with a 38kHz carrier for infrared transmission medium
- Minimum transmission word
  - Four words for the HT12E
  - One word for the HT12A
- Built-in oscillator needs only 5% resistor
- Data code has positive polarity
- Minimal external components
- HT12A/E: 18-pin DIP/20-pin SOP package

Applications

- Burglar alarm system
- Smoke and fire alarm system
- Garage door controllers
- Car door controllers
- Car alarm system
- Security system
- Cordless telephones
- Other remote control systems

General Description

The 2^{12} encoders are a series of CMOS LSIs for remote control system applications. They are capable of encoding information which consists of N address bits and 12–N data bits. Each address/data input can be set to one of the two logic states. The programmed addresses/data are transmitted together with the header bits via an RF or an infrared transmission medium upon receipt of a trigger signal. The capability to select a $\overline{TE}$ trigger on the HT12E or a DATA trigger on the HT12A further enhances the application flexibility of the 2^{12} series of encoders. The HT12A additionally provides a 38kHz carrier for infrared systems.

Selection Table

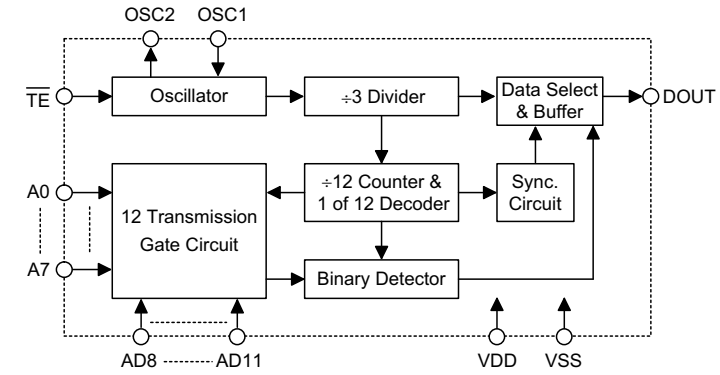
Function / Part No.	Address No.	Address/ Data No.	Data No.	Oscillator	Trigger	Package	Carrier Output	Negative Polarity
HT12A	8	0	4	455kHz resonator	D8~D11	18 DIP 20 SOP	38kHz	No
HT12E	8	4	0	RC oscillator	$\overline{TE}$	18 DIP 20 SOP	No	No

Note: Address/Data represents pins that can be address or data according to the decoder requirement.

Block Diagram

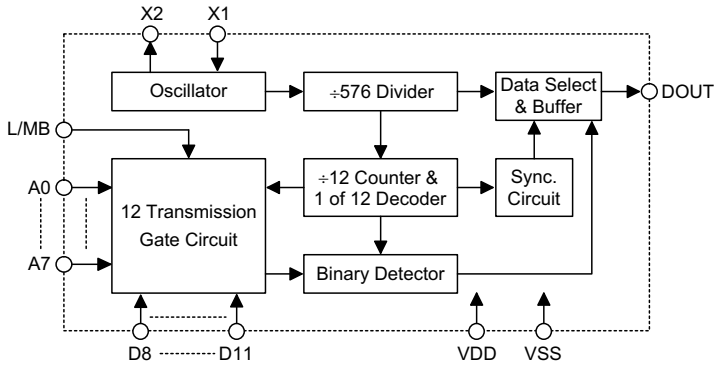
$\overline{TE}$ trigger

HT12E



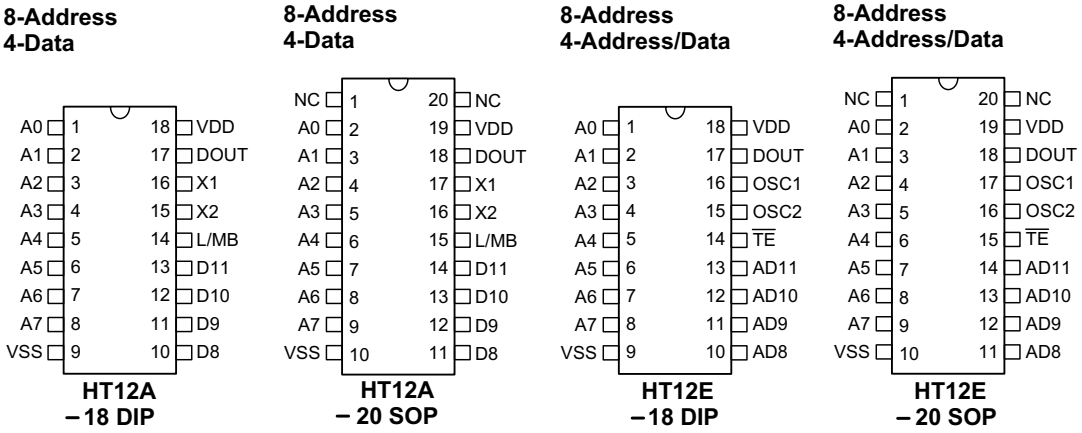
DATA trigger

HT12A



Note: The address data pins are available in various combinations (refer to the address/data table).

Pin Assignment



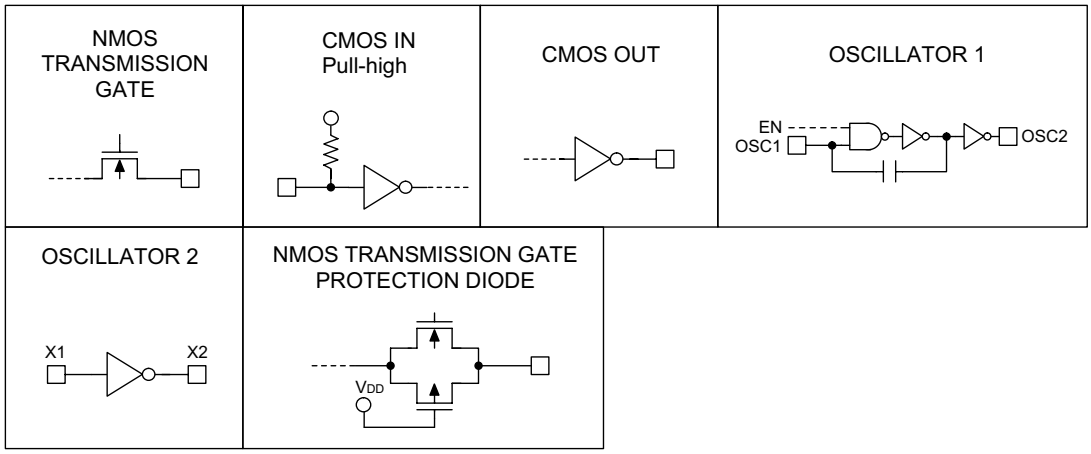
Pin Description

Pin Name	I/O	Internal Connection	Description
A0~A7	I	CMOS IN Pull-high (HT12A) NMOS TRANSMISSION GATE PROTECTION DIODE (HT12E)	Input pins for address A0~A7 setting These pins can be externally set to VSS or left open
AD8~AD11	I	NMOS TRANSMISSION GATE PROTECTION DIODE (HT12E)	Input pins for address/data AD8~AD11 setting These pins can be externally set to VSS or left open
D8~D11	I	CMOS IN Pull-high	Input pins for data D8~D11 setting and transmission enable, active low These pins should be externally set to VSS or left open (see Note)
DOUT	O	CMOS OUT	Encoder data serial transmission output
L/MB	I	CMOS IN Pull-high	Latch/Momentary transmission format selection pin: Latch: Floating or VDD Momentary: VSS

Pin Name	I/O	Internal Connection	Description
$\overline{TE}$	I	CMOS IN Pull-high	Transmission enable, active low (see Note)
OSC1	I	OSCILLATOR 1	Oscillator input pin
OSC2	O	OSCILLATOR 1	Oscillator output pin
X1	I	OSCILLATOR 2	455kHz resonator oscillator input
X2	O	OSCILLATOR 2	455kHz resonator oscillator output
VSS	I	—	Negative power supply, grounds
VDD	I	—	Positive power supply

Note: D8~D11 are all data input and transmission enable pins of the HT12A.
$\overline{TE}$ is a transmission enable pin of the HT12E.

Approximate internal connections



Absolute Maximum Ratings

Supply Voltage (HT12A) .....–0.3V to 5.5V
Input Voltage.....V_{SS}–0.3 to V_{DD}+0.3V
Operating Temperature.....–20°C to 75°C
Supply Voltage (HT12E) .....–0.3V to 13V
Storage Temperature.....–50°C to 125°C

Note: These are stress ratings only. Stresses exceeding the range specified under "Absolute Maximum Ratings" may cause substantial damage to the device. Functional operation of this device at other conditions beyond those listed in the specification is not implied and prolonged exposure to extreme conditions may affect device reliability.

Electrical Characteristics

HT12A Ta=25°C

Symbol	Parameter	Test Conditions		Min.	Typ.	Max.	Unit
		V_{DD}	Conditions				
V_{DD}	Operating Voltage	—	—	2.4	3	5	V
I_{STB}	Standby Current	3V	Oscillator stops	—	0.1	1	μA
		5V		—	0.1	1	μA
I_{DD}	Operating Current	3V	No load f_{OSC}=455kHz	—	200	400	μA
		5V		—	400	800	μA
I_{DOUT}	Output Drive Current	5V	V_{OH}=0.9V_{DD} (Source)	–1	–1.6	—	mA
			V_{OL}=0.1V_{DD} (Sink)	2	3.2	—	mA
V_{IH}	"H" Input Voltage	—	—	0.8V_{DD}	—	V_{DD}	V
V_{IL}	"L" Input Voltage	—	—	0	—	0.2V_{DD}	V
R_{DATA}	D8~D11 Pull-high Resistance	5V	V_{DATA}=0V	—	150	300	kΩ

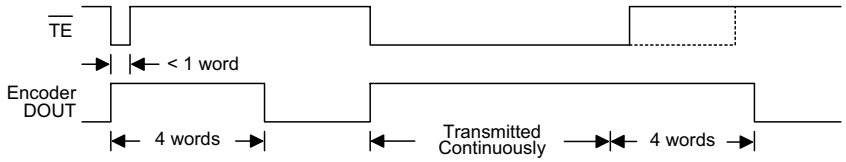
HT12E Ta=25°C

Symbol	Parameter	Test Conditions		Min.	Typ.	Max.	Unit
		V_{DD}	Conditions				
V_{DD}	Operating Voltage	—	—	2.4	5	12	V
I_{STB}	Standby Current	3V	Oscillator stops	—	0.1	1	μA
		12V		—	2	4	μA
I_{DD}	Operating Current	3V	No load f_{OSC}=3kHz	—	40	80	μA
		12V		—	150	300	μA
I_{DOUT}	Output Drive Current	5V	V_{OH}=0.9V_{DD} (Source)	–1	–1.6	—	mA
			V_{OL}=0.1V_{DD} (Sink)	1	1.6	—	mA
V_{IH}	"H" Input Voltage	—	—	0.8V_{DD}	—	V_{DD}	V
V_{IL}	"L" Input Voltage	—	—	0	—	0.2V_{DD}	V
f_{OSC}	Oscillator Frequency	5V	R_{OSC}=1.1MΩ	—	3	—	kHz
$R_{\overline{TE}}$	$\overline{TE}$ Pull-high Resistance	5V	$V_{\overline{TE}}$=0V	—	1.5	3	MΩ

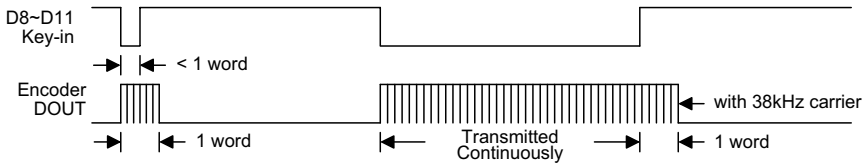
Functional Description

Operation

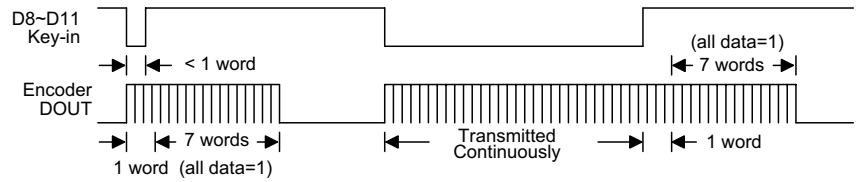
The 2^{12} series of encoders begin a 4-word transmission cycle upon receipt of a transmission enable ($\overline{TE}$ for the HT12E or D8~D11 for the HT12A, active low). This cycle will repeat itself as long as the transmission enable ($\overline{TE}$ or D8~D11) is held low. Once the transmission enable returns high the encoder output completes its final cycle and then stops as shown below.



Transmission timing for the HT12E



Transmission timing for the HT12A (L/MB=Floating or VDD)

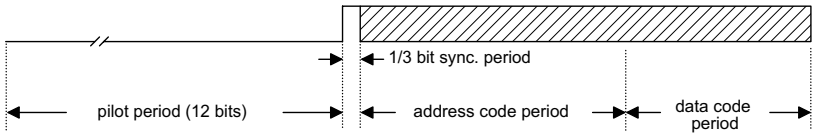


Transmission timing for the HT12A (L/MB=VSS)

Information word

If L/MB=1 the device is in the latch mode (for use with the latch type of data decoders). When the transmission enable is removed during a transmission, the DOUT pin outputs a complete word and then stops. On the other hand, if L/MB=0 the device is in the momentary mode (for use with the momentary type of data decoders). When the transmission enable is removed during a transmission, the DOUT outputs a complete word and then adds 7 words all with the "1" data code.

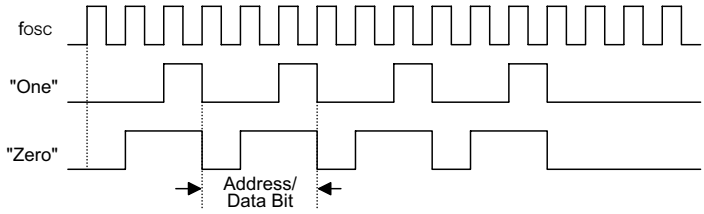
An information word consists of 4 periods as illustrated below.



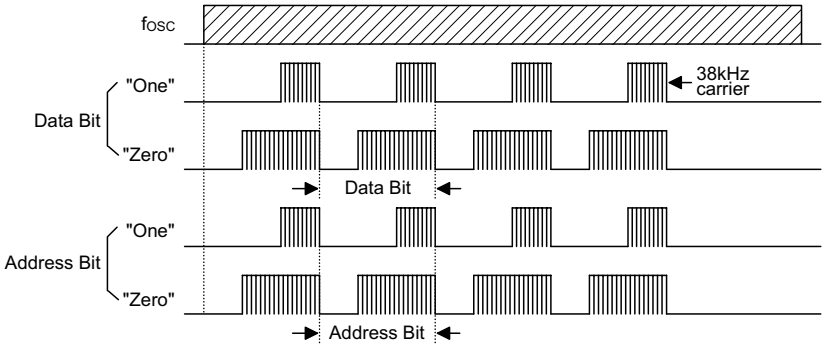
Composition of information

Address/data waveform

Each programmable address/data pin can be externally set to one of the following two logic states as shown below.



Address/Data bit waveform for the HT12E



Address/Data bit waveform for the HT12A

The address/data bits of the HT12A are transmitted with a 38kHz carrier for infrared remote controller flexibility.

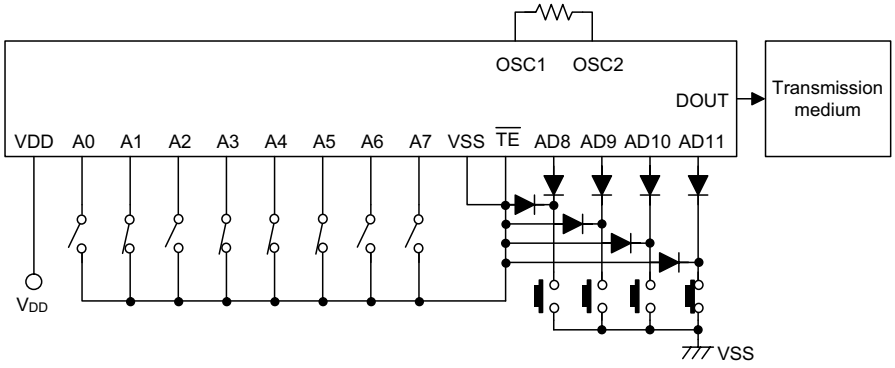
Address/data programming (preset)

The status of each address/data pin can be individually pre-set to logic "high" or "low". If a transmission-enable signal is applied, the encoder scans and transmits the status of the 12 bits of address/data serially in the order A0 to AD11 for the HT12E encoder and A0 to D11 for the HT12A encoder.

During information transmission these bits are transmitted with a preceding synchronization bit. If the trigger signal is not applied, the chip enters the standby mode and consumes a reduced current of less than 1μA for a supply voltage of 5V.

Usual applications preset the address pins with individual security codes using DIP switches or PCB wiring, while the data is selected by push buttons or electronic switches.

The following figure shows an application using the HT12E:



The transmitted information is as shown:

Pilot & Sync.	A0 1	A1 0	A2 1	A3 0	A4 0	A5 0	A6 1	A7 1	AD8 1	AD9 1	AD10 1	AD11 0

Address/Data sequence

The following provides the address/data sequence table for various models of the 2^{12} series of encoders. The correct device should be selected according to the individual address and data requirements.

Part No.	Address/Data Bits											
	0	1	2	3	4	5	6	7	8	9	10	11
HT12A	A0	A1	A2	A3	A4	A5	A6	A7	D8	D9	D10	D11
HT12E	A0	A1	A2	A3	A4	A5	A6	A7	AD8	AD9	AD10	AD11

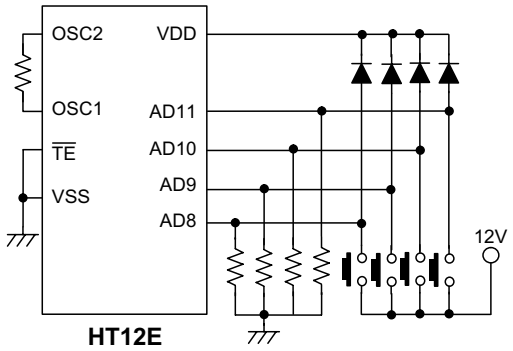
Transmission enable

For the HT12E encoders, transmission is enabled by applying a low signal to the $\overline{TE}$ pin. For the HT12A encoders, transmission is enabled by applying a low signal to one of the data pins D8~D11.

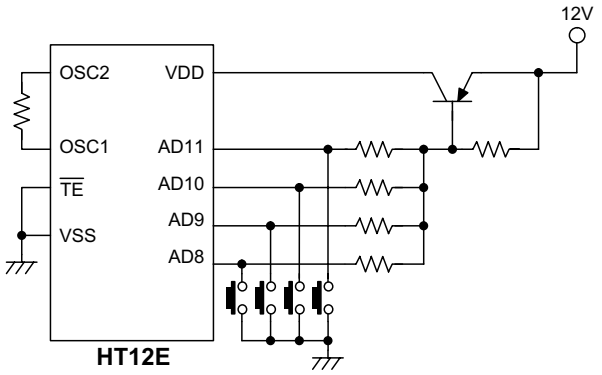
Two erroneous HT12E application circuits

The HT12E must follow closely the application circuits provided by Holtek (see the "Application circuits").

- Error: AD8~AD11 pins input voltage > V_{DD}+0.3V

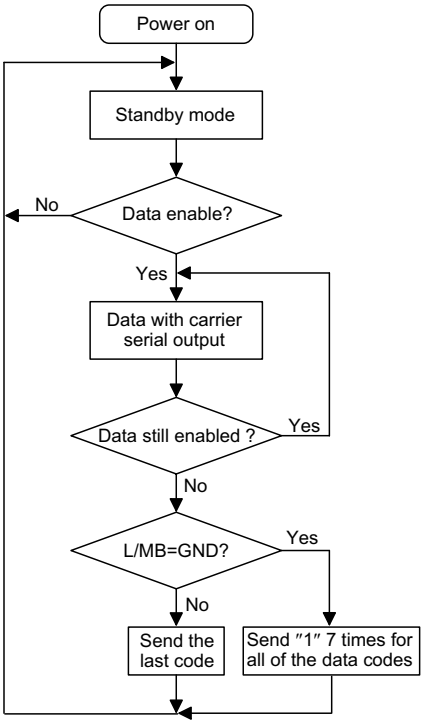


• Error: The IC′s power source is activated by pins AD8~AD11

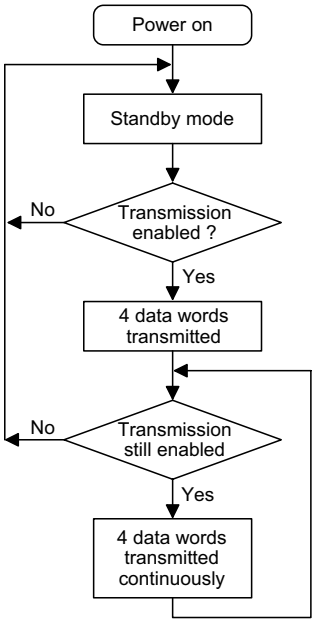


Flowchart

• HT12A

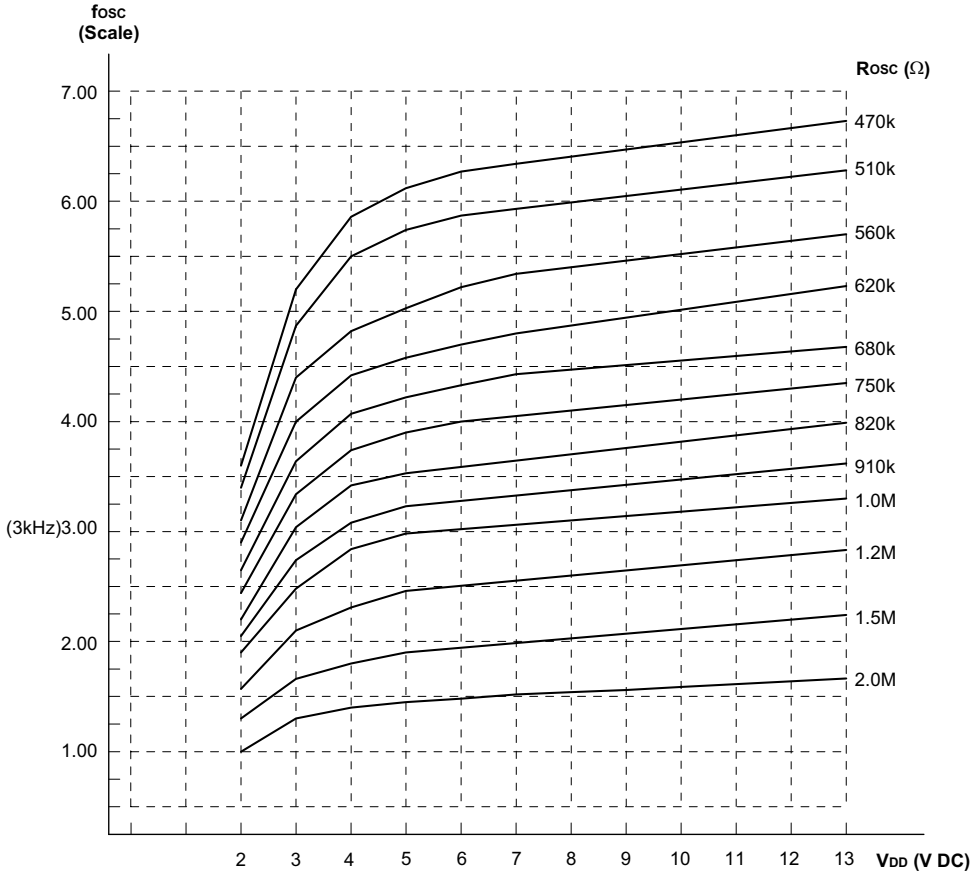


• HT12E



Note: D8~D11 are transmission enables of the HT12A.
$\overline{TE}$ is the transmission enable of the HT12E.

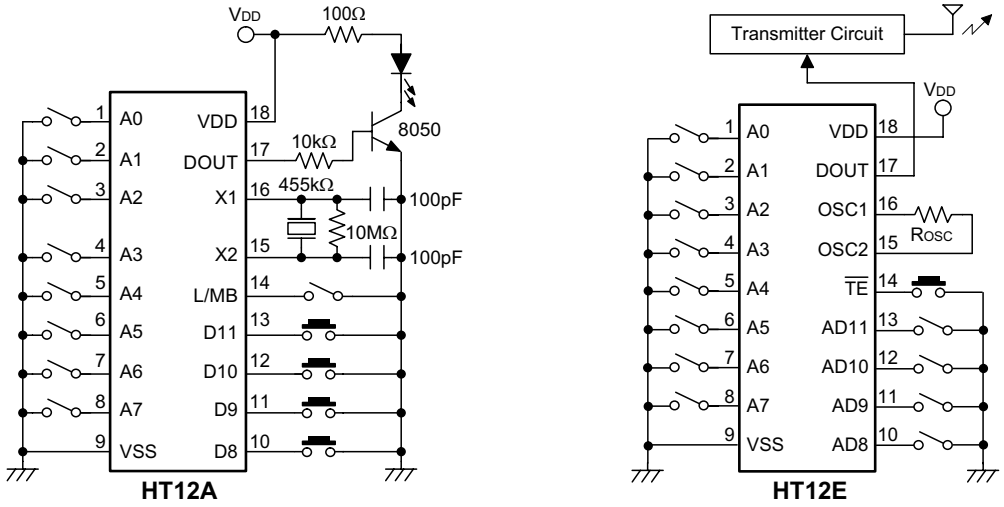
Oscillator frequency vs supply voltage



The recommended oscillator frequency is f_{OSCD} (decoder) $\cong 50\ f_{OSCE}$ (HT12E encoder)

$$\cong \frac{1}{3} f_{OSCE} \text{ (HT12A encoder)}$$

Application Circuits



Note: Typical infrared diode: EL-1L2 (KODENSHI CORP.)
Typical RF transmitter: JR-220 (JUWA CORP.)

Holtek Semiconductor Inc. (Headquarters)
No.3, Creation Rd. II, Science-based Industrial Park, Hsinchu, Taiwan, R.O.C.
Tel: 886-3-563-1999
Fax: 886-3-563-1189

Holtek Semiconductor Inc. (Taipei Office)
11F, No.576, Sec.7 Chung Hsiao E. Rd., Taipei, Taiwan, R.O.C.
Tel: 886-2-2782-9635
Fax: 886-2-2782-9636
Fax: 886-2-2782-7128 (International sales hotline)

Holtek Semiconductor (Hong Kong) Ltd.
RM.711, Tower 2, Cheung Sha Wan Plaza, 833 Cheung Sha Wan Rd., Kowloon, Hong Kong
Tel: 852-2-745-8288
Fax: 852-2-742-8657

Holtek Semiconductor (Shanghai) Ltd.
7th Floor, Building 2, No.889, Yi Shan Rd., Shanghai, China
Tel: 021-6485-5560
Fax: 021-6485-0313

Holmate Technology Corp.
48531 Warm Springs Boulevard, Suite 413, Fremont, CA 94539
Tel: 510-252-9880
Fax: 510-252-9885

Crystalfontz America, Inc.
15611 East Washington Road
Valleyford, WA 99036

Phone: (509) 291-3514
Fax: (509) 291-3345

http://www.crystalfontz.com
email: sales@crystalfontz.com

Crystalfontz America, Inc.

CUSTOMER		
MODEL	**CFAH1601A-AGB-JP**	
APPROVAL	**BY:**	**DATA:**

SALES BY	APPROVED BY	CHECKED BY	PREPARED BY

Crystalfontz America, Inc.
15611 East Washington Road
Valleyford, WA 99036-9747
Phone: (509) 291-3514
Fax: (509) 291-3345
e-mail: sales@crystalfontz.com

Crystalfontz America, Inc.
15611 East Washington Road
Valleyford, WA 99036

Phone: (509) 291-3514
Fax: (509) 291-3345

http://www.crystalfontz.com
email: sales@crystalfontz.com

Contents

Crystalfontz America, Inc.
15611 East Washington Road
Valleyford, WA 99036

Phone: (509) 291-3514
Fax: (509) 291-3345

http://www.crystalfontz.com
email: sales@crystalfontz.com

1.Module Classification Information

CFA H 1601 A A G B JP
① ② ③ ④ ⑤ ⑥ ⑦ ⑧

①	Brand: CRYSTALFONTZ AMERICA, INCORPORATED		
②	Display Type: **H➔Character Type**, G➔Graphic Type		
③	Display's logical dimensions: **16** columns by **01** lines		
④	Model serials no.		
⑤	Backlight Type:	N➔Without backlight B➔EL, Blue green D➔EL, Green W➔EL, White F➔CCFL, White T➔LED, White	Y➔LED, Yellow Green **A➔LED, Amber** R➔LED, Red O➔LED, Orange G➔LED, Green
⑥	LCD Mode:	B➔TN Positive, Gray N➔TN Negative, **G➔STN Positive, Gray** Y➔STN Positive, Yellow Green	M➔STN Negative, Blue F➔FSTN Positive T➔FSTN Negative
⑦	LCD Polarizer Type, Temperature range, Viewing direction:	A➔Reflective, N.T, 6:00 D➔Reflective, N.T, 12:00 G➔Reflective, W. T, 6:00 J➔Reflective, W. T, 12:00 **B➔Transflective, N.T,6:00** E➔Transflective, N.T.12:00	H➔Transflective, W.T,6:00 K➔Transflective, W.T,12:00 C➔Transmissive, N.T,6:00 F➔Transmissive, N.T,12:00 I➔Transmissive, W. T, 6:00 L➔Transmissive, W.T,12:00
⑧	Special Code:	**JP➔English and Japanese standard font**	

2.Precautions in use of LCD Modules

(1)Avoid applying excessive shocks to the module or making any alterations or modifications to it.

(2)Don't make extra holes on the printed circuit board, modify its shape or change the components of LCD module.

(3)Don't disassemble the LCM.

(4)Don't operate it above the absolute maximum rating.

(5)Don't drop, bend or twist LCM.

(6)Soldering: only to the I/O terminals.

(7)Storage: please storage in anti-static electricity container and clean environment.

3.General Specification

Item	Dimension	Unit
Number of Characters	16 characters x 1 Lines	—
Module dimension	80.0 x 36.0 x 13.2(MAX)	mm
View area	66.0 x 16.0	mm
Active area	59.62 x 6.56	mm
Dot size	0.55 x 0.75	mm
Dot pitch	0.63 x 0.83	mm
Character size	3.07 x 6.56	mm
Character pitch	3.77 x 6.56	mm
LCD type	STN, Positive, Transflective,Gray	
Duty	1/16	
View direction	6 o'clock	
Backlight Type	LED Amber	

4.Absolute Maximum Ratings

Item	Symbol	Min	Typ	Max	Unit
Operating Temperature	T_{OP}	0	—	+50	℃
Storage Temperature	T_{ST}	-10	—	+60	℃
Input Voltage	V_I	V_{SS}	—	V_{DD}	V
Supply Voltage For Logic	V_{DD}-V_{SS}	-0.3	—	7	V
Supply Voltage For LCD	V_{DD}-V_0	-0.3	—	13	V

5.Electrical Characteristics

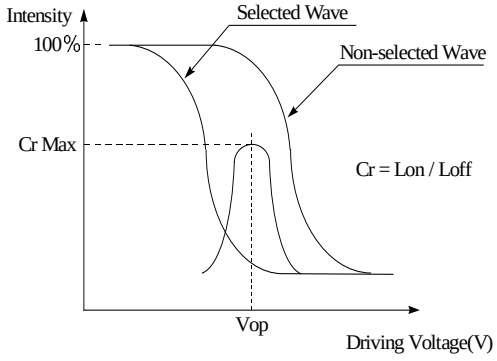
Item	Symbol	Condition	Min	Typ	Max	Unit
Supply Voltage For Logic	V_{DD}-V_{SS}	—	4.5	—	5.5	V
Supply Voltage For LCD	V_{DD}-V_0	Ta=0℃	—	—	4.8	V
		Ta=25℃	—	4.5	—	V
		Ta=50℃	4.2	—	—	V
Input High Volt.	V_{IH}	—	2.2	—	V_{DD}	V
Input Low Volt.	V_{IL}	—	—	—	0.6	V
Output High Volt.	V_{OH}	—	2.4	—	—	V
Output Low Volt.	V_{OL}	—	—	—	0.4	V
Supply Current	I_{DD}	V_{DD}=5V	—	1.2	—	mA

Crystalfontz America, Inc.
15611 East Washington Road
Valleyford, WA 99036
Phone: (509) 291-3514
Fax: (509) 291-3345
http://www.crystalfontz.com
email: sales@crystalfontz.com

6.Optical Characteristics

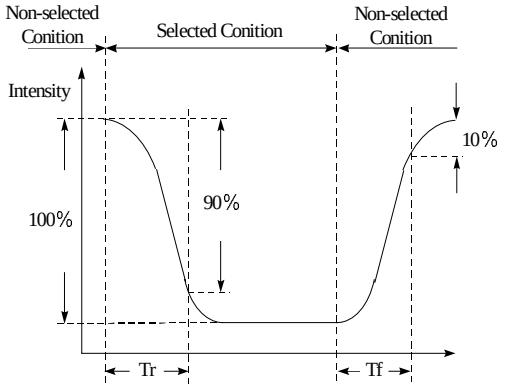
Item	Symbol	Condition	Min	Typ	Max	Unit
View Angle	(V) θ	CR≧2	10	—	105	deg
	(H) φ	CR≧2	-30	—	30	deg
Contrast Ratio	CR	—	—	3	—	—
Response Time	T rise	—	—	150	200	ms
	T fall	—	—	150	200	ms

Definition of Operation Voltage (Vop)



[positive type]

Definition of Response Time (Tr , Tf)



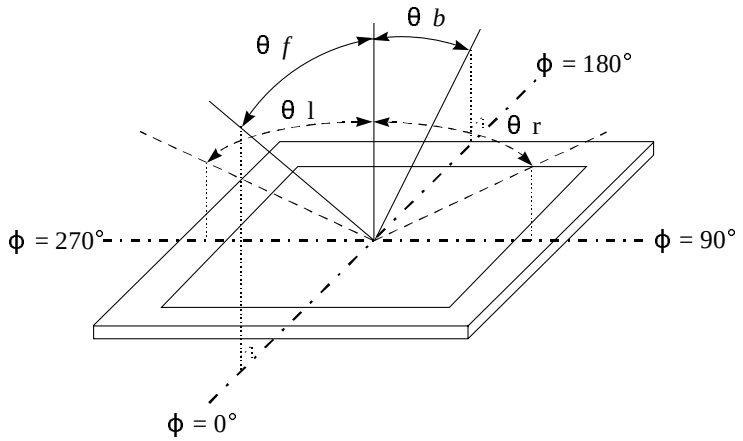
[positive type]

Conditions :

Operating Voltage : Vop Viewing Angle(θ , φ) : 0°, 0°

Frame Frequency : 64 HZ Driving Waveform : 1/N duty , 1/a bias

Definition of viewing angle(CR≧2)



7.Interface Pin Function

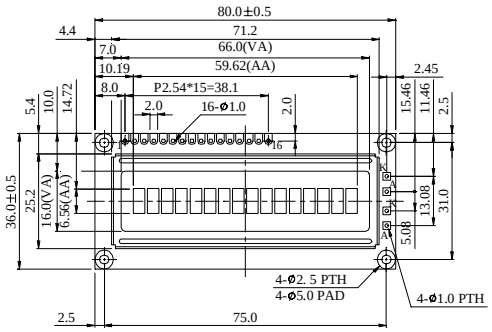
Pin No.	Symbol	Level	Description
1	V_{SS}	0V	Ground
2	V_{DD}	5.0V	Supply Voltage for logic
3	VO	(Variable)	Operating voltage for LCD
4	RS	H/L	H: DATA, L: Instruction code
5	R/W	H/L	H: Read(MPU→Module) L: Write(MPU→Module)
6	E	H,H→L	Chip enable signal
7	DB0	H/L	Data bit 0
8	DB1	H/L	Data bit 1
9	DB2	H/L	Data bit 2
10	DB3	H/L	Data bit 3
11	DB4	H/L	Data bit 4
12	DB5	H/L	Data bit 5
13	DB6	H/L	Data bit 6
14	DB7	H/L	Data bit 7
15	A	—	LED +
16	K	—	LED—

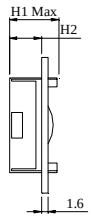
Crystalfontz America, Inc.
15611 East Washington Road
Valleyford, WA 99036

Phone: (509) 291-3514
Fax: (509) 291-3345

http://www.crystalfontz.com
email: sales@crystalfontz.com

8.Contour Drawing &Block Diagram



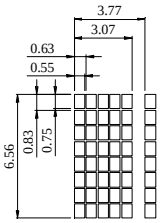


LED B/L

LED-H/L B/L

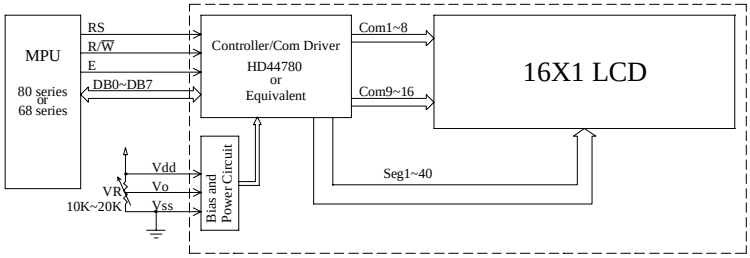
	High	Low
H1	13.2	
H2	8.6	

PIN NO.	SYMBOL
1	Vss
2	Vdd
3	Vo
4	RS
5	R/$\overline{W}$
6	E
7	DB0
8	DB1
9	DB2
10	DB3
11	DB4
12	DB5
13	DB6
14	DB7
15	A/Vee
16	K



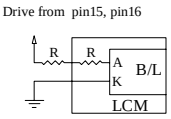
DOT SIZE

The non-specified tolerance of dimension is ±0.3mm.



External contrast adjustment.

LED B/L Drive Method



Character located	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
DDRAM address	00	01	02	03	04	05	06	07	40	41	42	43	44	45	46	47

2-line display mode.

9.Function Description

The LCD display Module is built in a LSI controller, the controller has two 8-bit registers, an instruction register (IR) and a data register (DR).
The IR stores instruction codes, such as display clear and cursor shift, and address information for display data RAM (DDRAM) and character generator (CGRAM). The IR can only be written from the MPU. The DR temporarily stores data to be written or read from DDRAM or CGRAM. When address information is written into the IR, then data is stored into the DR from DDRAM or CGRAM. By the register selector (RS) signal, these two registers can be selected.

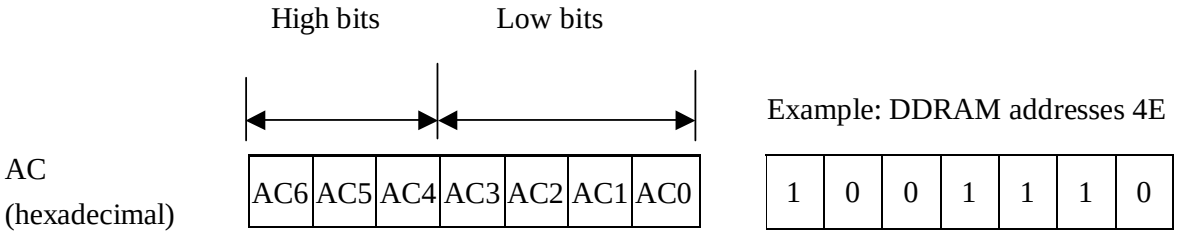
RS	R/W	Operation
0	0	IR write as an internal operation (display clear, etc.)
0	1	Read busy flag (DB7) and address counter (DB0 to DB7)
1	0	Write data to DDRAM or CGRAM (DR to DDRAM or CGRAM)
1	1	Read data from DDRAM or CGRAM (DDRAM or CGRAM to DR)

Busy Flag (BF)
When the busy flag is 1, the controller LSI is in the internal operation mode, and the next instruction will not be accepted. When RS=0 and R/W=1, the busy flag is output to DB7. The next instruction must be written after ensuring that the busy flag is 0.

Address Counter (AC)
The address counter (AC) assigns addresses to both DDRAM and CGRAM

Display Data RAM (DDRAM)

This DDRAM is used to store the display data represented in 8-bit character codes. Its extended capacity is 80×8 bits or 80 characters. Below figures are the relationships between DDRAM addresses and positions on the liquid crystal display.



Display position DDRAM address

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
00	01	02	03	04	05	06	07	40	41	42	43	44	45	46	47

1-Line by 16-Character Display

Character Generator ROM (CGROM)

The CGROM generate 5×8 dot or 5×10 dot character patterns from 8-bit character codes. See Table 2.

Character Generator RAM (CGRAM)

In CGRAM, the user can rewrite character by program. For 5×8 dots, eight character patterns can be written, and for 5×10 dots, four character patterns can be written.

Write into DDRAM the character code at the addresses shown as the left column of table 1. To show the character patterns stored in CGRAM.

Crystalfontz America, Inc.
15611 East Washington Road
Valleyford, WA 99036

Phone: (509) 291-3514
Fax: (509) 291-3345

http://www.crystalfontz.com
email: sales@crystalfontz.com

Relationship between CGRAM Addresses, Character Codes (DDRAM) and Character patterns

Table 1.

For 5 * 8 dot character patterns

Character Codes (DDRAM data)		CGRAM Address		Character Patterns (CGRAM data)	
7 6 5 4 3 2 1 0		5 4 3 2 1 0		7 6 5 4 3 2 1 0	
High Low		High Low		High Low	
0 0 0 0 * 0 0 0		0 0 0	0 0 0	* * *	Character pattern(1)
			0 0 1	* * *	
			0 1 0	* * *	
			0 1 1	* * *	
			1 0 0	* * *	
			1 0 1	* * *	
			1 1 0	* * *	
			1 1 1	* * *	Cursor pattern
0 0 0 0 * 0 0 1		0 0 1	0 0 0	* * *	Character pattern(2)
			0 0 1	* * *	
			0 1 0	* * *	
			0 1 1	* * *	
			1 0 0	* * *	
			1 0 1	* * *	
			1 1 0	* * *	
			1 1 1	* * *	Cursor pattern
			0 0 0	* * *	
			0 0 1		
0 0 0 0 * 1 1 1		1 1 1	1 0 0		
			1 0 1		
			1 1 0		
			1 1 1	* * *	

For 5 * 10 dot character patterns

Character Codes (DDRAM data)		CGRAM Address		Character Patterns (CGRAM data)		
7 6 5 4 3 2 1 0		5 4 3 2 1 0		7 6 5 4 3 2 1 0		
High Low		High Low		High Low		
0 0 0 0 * 0 0 0		0 0	0 0 0 0	* * *	0 0 0 0 0	Character pattern
			0 0 0 1	* * *	0 0 0 0 0	
			0 0 1 0	* * *	0	
			0 0 1 1	* * *	0 0	
			0 1 0 0	* * *	0 0 0	
			0 1 0 1	* * *	0 0 0	
			0 1 1 0	* * *		
			0 1 1 1	* * *	0 0 0 0	
			1 0 0 0	* * *	0 0 0 0	
			1 0 0 1	* * *	0 0 0 0	
			1 0 1 0	* * *	0 0 0 0 0	Cursor pattern
			1 1 1 1	* * *	* * * * *	

■ : " High "

Crystalfontz America, Inc.
15611 East Washington Road
Valleyford, WA 99036

Phone: (509) 291-3514
Fax: (509) 291-3345

http://www.crystalfontz.com
email: sales@crystalfontz.com

10.Character Generator ROM Pattern

Table.2

Upper 4 bit / Lower 4 bit	LLLL	LLLH	LLHL	LLHH	LHLL	LHLH	LHHL	LHHH	HLLL	HLLH	HLHL	HLHH	HHLL	HHLH	HHHL	HHHH
LLLL	CG RAM (1)															
LLLH	(2)															
LLHL	(3)															
LLHH	(4)															
LHLL	(5)															
LHLH	(6)															
LHHL	(7)															
LHHH	(8)															
HLLL	(1)															
HLLH	(2)															
HLHL	(3)															
HLHH	(4)															
HHLL	(5)															
HHLH	(6)															
HHHL	(7)															
HHHH	(8)															

Crystalfontz America, Inc.
15611 East Washington Road
Valleyford, WA 99036
Phone: (509) 291-3514
Fax: (509) 291-3345
http://www.crystalfontz.com
email: sales@crystalfontz.com

11.Instruction Table

Instruction	Instruction Code										Description	Execution time (fosc=270Khz)
	RS	R/W	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0		
Clear Display	0	0	0	0	0	0	0	0	0	1	Write “00H” to DDRAM and set DDRAM address to “00H” from AC	1.53ms
Return Home	0	0	0	0	0	0	0	0	1	—	Set DDRAM address to “00H” from AC and return cursor to its original position if shifted. The contents of DDRAM are not changed.	1.53ms
Entry Mode Set	0	0	0	0	0	0	0	1	I/D	SH	Assign cursor moving direction and enable the shift of entire display.	39 μs
Display ON/OFF Control	0	0	0	0	0	0	1	D	C	B	Set display (D), cursor (C), and blinking of cursor (B) on/off control bit.	39 μs
Cursor or Display Shift	0	0	0	0	0	1	S/C	R/L	—	—	Set cursor moving and display shift control bit, and the direction, without changing of DDRAM data.	39 μs
Function Set	0	0	0	0	1	DL	N	F	—	—	Set interface data length (DL:8-bit/4-bit), numbers of display line (N:2-line/1-line)and, display font type (F:5×11 dots/5× 8 dots)	39 μs
Set CGRAM Address	0	0	0	1	AC5	AC4	AC3	AC2	AC1	AC0	Set CGRAM address in address counter.	39 μs
Set DDRAM Address	0	0	1	AC6	AC5	AC4	AC3	AC2	AC1	AC0	Set DDRAM address in address counter.	39 μs
Read Busy Flag and Address	0	1	BF	AC6	AC5	AC4	AC3	AC2	AC1	AC0	Whether during internal operation or not can be known by reading BF. The contents of address counter can also be read.	0 μs
Write Data to RAM	1	0	D7	D6	D5	D4	D3	D2	D1	D0	Write data into internal RAM (DDRAM/CGRAM).	43 μs
Read Data from RAM	1	1	D7	D6	D5	D4	D3	D2	D1	D0	Read data from internal RAM (DDRAM/CGRAM).	43 μs

＊ ”—”：don’t care

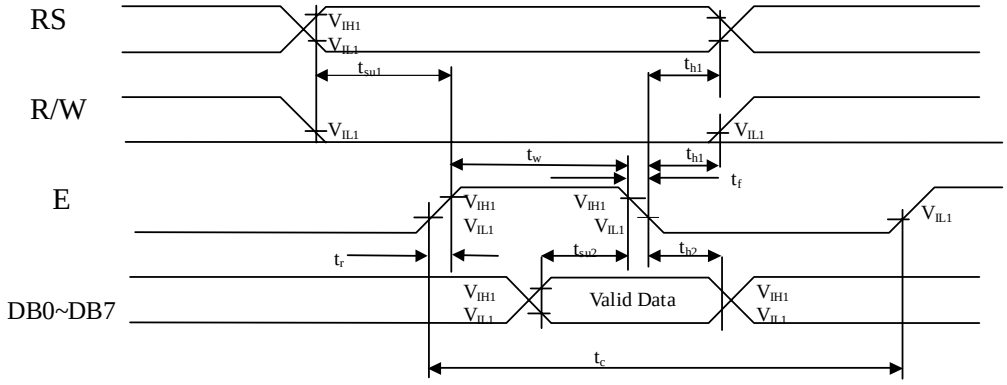
Crystalfontz America, Inc.
15611 East Washington Road
Valleyford, WA 99036

Phone: (509) 291-3514
Fax: (509) 291-3345

http://www.crystalfontz.com
email: sales@crystalfontz.com

12.Timing Characteristics

12.1 Write Operation

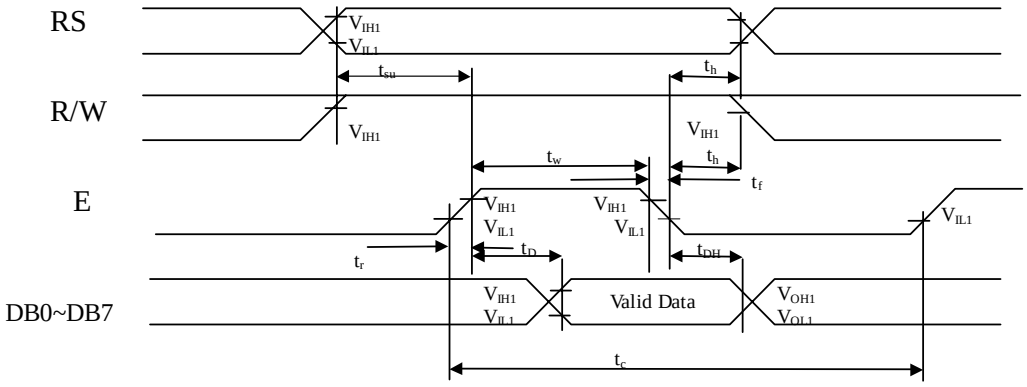


(V_{DD}=4.5V~5.5V , Ta=-30~+85℃)

Mode	Characteristic	Symbol	Min.	Typ.	Max.	Unit
Write Mode	**E cycle Time**	**tc**	**500**	—	—	**ns**
	E Rise/Fall Time	**t_R, t_F**	—	—	**20**	
	E Pulse Width (High, Low)	**tw**	**230**	—	—	
	R/W and RS Setup Time	**tsu1**	**40**	—	—	
	R/W and RS Hold Time	**t_{H1}**	**10**	—	—	
	Data Setup Time	**tsu2**	**80**	—	—	
	Data Hold Time	**t_{H2}**	**10**	—	—	

Crystalfontz America, Inc.
15611 East Washington Road
Valleyford, WA 99036
Phone: (509) 291-3514
Fax: (509) 291-3345
http://www.crystalfontz.com
email: sales@crystalfontz.com

12.2 Read Operation



(V_{DD}=4.5V~5.5V , Ta=-30~+85℃)

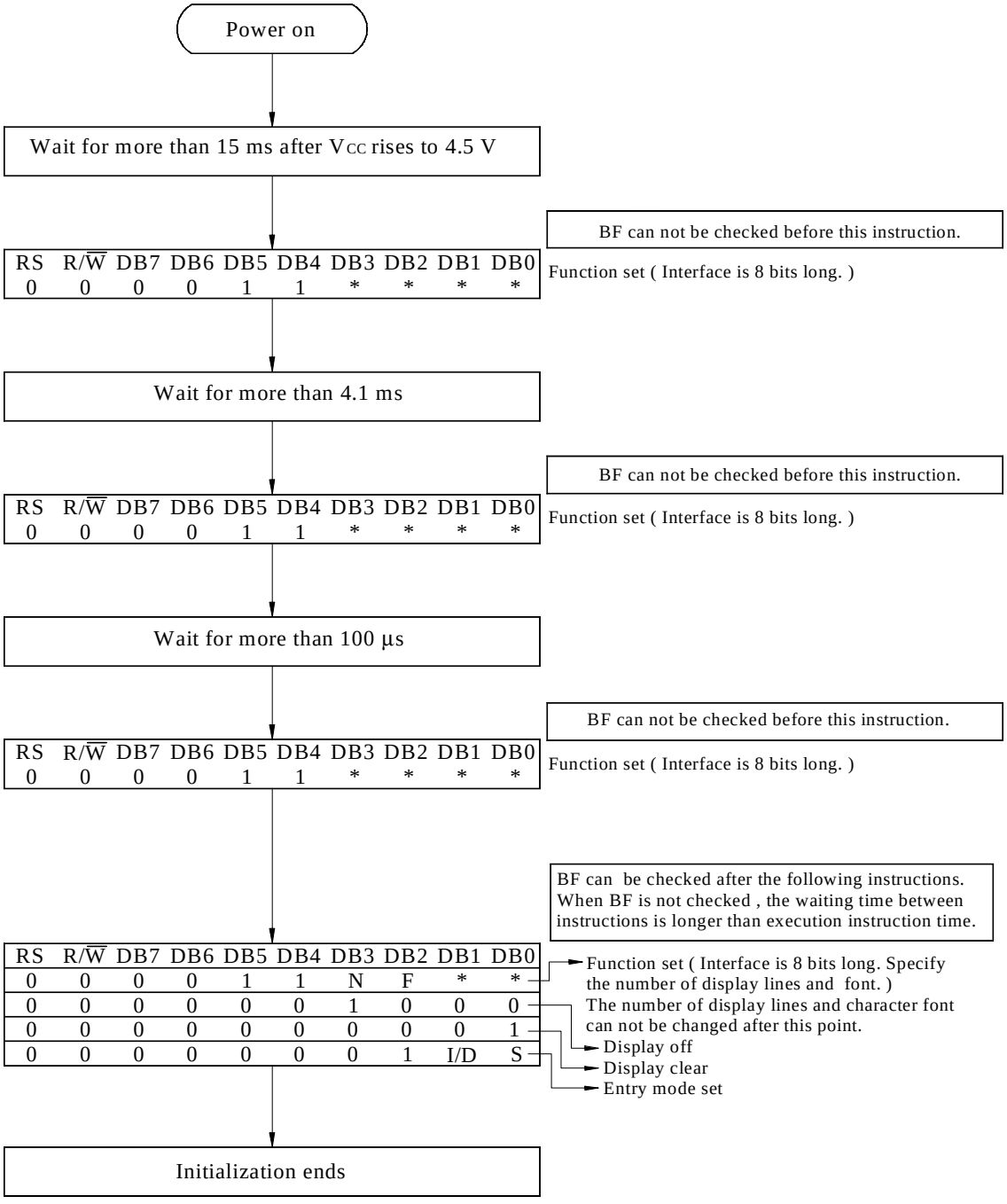
Mode	Characteristic	Symbol	Min.	Typ.	Max.	Unit
Read Mode	**E cycle Time**	**tc**	**500**	—	—	**ns**
	E Rise/Fall Time	**t_R, t_F**	—	—	**20**	
	E Pulse Width (High, Low)	**tw**	**230**	—	—	
	R/W and RS Setup Time	**tsu**	**40**	—	—	
	R/W and RS Hold Time	**t_H**	**10**	—	—	
	Data Output Delay Time	**t_D**	—	—	**120**	
	Data Hold Time	**t_{DH}**	**5**	—	—	

Crystalfontz America, Inc.
15611 East Washington Road
Valleyford, WA 99036

Phone: (509) 291-3514
Fax: (509) 291-3345

http://www.crystalfontz.com
email: sales@crystalfontz.com

13.Initializing of LCM

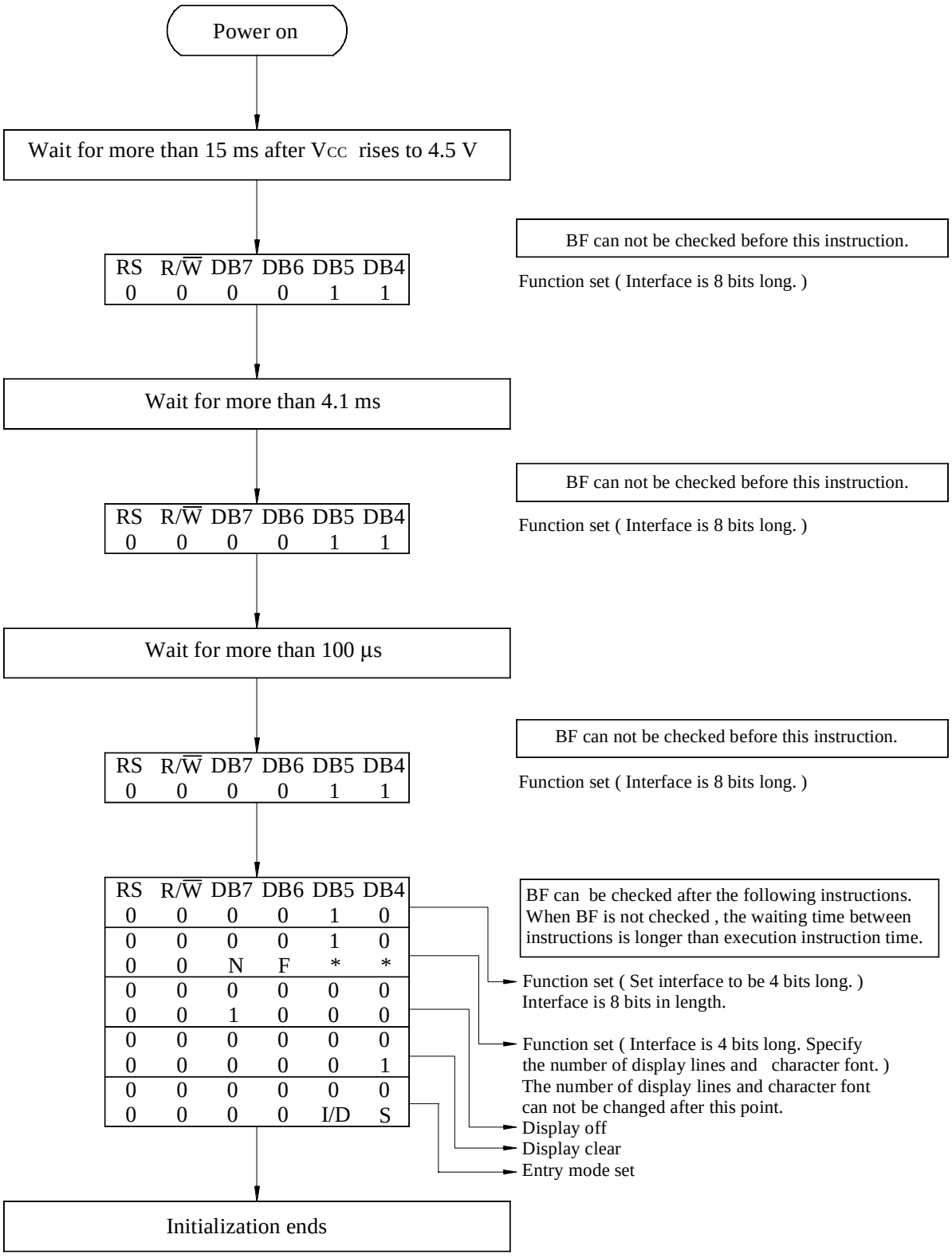


8-Bit Ineterface

Crystalfontz America, Inc.
15611 East Washington Road
Valleyford, WA 99036

Phone: (509) 291-3514
Fax: (509) 291-3345

http://www.crystalfontz.com
email: sales@crystalfontz.com



4-Bit Ineterface

Crystalfontz America, Inc.
15611 East Washington Road
Valleyford, WA 99036

Phone: (509) 291-3514
Fax: (509) 291-3345

http://www.crystalfontz.com
email: sales@crystalfontz.com

14.Quality Assurance

Screen Cosmetic Criteria

Item	Defect	Judgment Criterion	Partition
1	Spots	A)Clear Size: d mm — Acceptable Qty in active area d ≦0.1 — Disregard 0.1<d≦0.2 — 6 0.2<d≦0.3 — 2 0.3<d — 0 Note: Including pin holes and defective dots which must be within one pixel size. B)Unclear Size: d mm — Acceptable Qty in active area d ≦0.2 — Disregard 0.2<d≦0.5 — 6 0.5<d≦0.7 — 2 0.7<d — 0	Minor
2	Bubbles in Polarizer	Size: d mm — Acceptable Qty in active area d≦0.3 — Disregard 0.3<d≦1.0 — 3 1.0<d≦1.5 — 1 1.5<d — 0	Minor
3	Scratch	In accordance with spots cosmetic criteria. When the light reflects on the panel surface, the scratches are not to be remarkable.	Minor
4	Allowable Density	Above defects should be separated more than 30mm each other.	Minor
5	Coloration	Not to be noticeable coloration in the viewing area of the LCD panels. Back-light type should be judged with back-light on state only.	Minor

15.Reliability

Content of Reliability Test

Environmental Test			
Test Item	**Content of Test**	**Test Condition**	**Applicable Standard**
High Temperature storage	Endurance test applying the high storage temperature for a long time.	60℃ 200hrs	——
Low Temperature storage	Endurance test applying the high storage temperature for a long time.	-10℃ 200hrs	——
High Temperature Operation	Endurance test applying the electric stress (Voltage & Current) and the thermal stress to the element for a long time.	50℃ 200hrs	——
Low Temperature Operation	Endurance test applying the electric stress under low temperature for a long time.	0℃ 200hrs	——
High Temperature/ Humidity Storage	Endurance test applying the high temperature and high humidity storage for a long time.	60℃,90%RH 96hrs	——
High Temperature/ Humidity Operation	Endurance test applying the electric stress (Voltage & Current) and temperature / humidity stress to the element for a long time.	50℃,90%RH 96hrs	——
Temperature Cycle	Endurance test applying the low and high temperature cycle. -10℃ 25℃ 60℃ 30min 5min 30min 1 cycle	-10℃/60℃ 10 cycles	——
Mechanical Test			
Vibration test	Endurance test applying the vibration during transportation and using.	10~22Hz→1.5mmp-p 22~500Hz→1.5G Total 0.5hrs	——
Shock test	Constructional and mechanical endurance test applying the shock during transportation.	50G Half sign wave 11 msedc 3 times of each direction	——
Atmospheric pressure test	Endurance test applying the atmospheric pressure during transportation by air.	115mbar 40hrs	——
Others			
Static electricity test	Endurance test applying the electric stress to the terminal.	VS=800V,RS=1.5kΩ CS=100pF 1 time	——

***Supply voltage for logic system=5V. Supply voltage for LCD system =Operating voltage at 25℃

16.Backlight Information

Specification

PARAMETER	SYMBOL	MIN	TYP	MAX	UNIT	TEST CONDITION
Supply Current	**ILED**	—	**130**	—	**mA**	**V=4.2V**
Supply Voltage	**V**	—	**4.2**	**4.6**	**V**	—
Reverse Voltage	**VR**	—	—	**8**	**V**	—
Luminous Intensity	**IV**	**60**	—	—	**CD/M^2**	**ILED=130mA**
Wave Length	**λp**	—	**610**	—	**nm**	**ILED=130mA**
Life Time	—	—	**100000**	—	**Hr.**	**V≦4.6V**
Color	**Amber**					

Crystalfontz America, Inc.
15611 East Washington Road
Valleyford, WA 99036

Phone: (509) 291-3514
Fax: (509) 291-3345

http://www.crystalfontz.com
email: sales@crystalfontz.com

Crystalfontz America, Inc.

CUSTOMER		
MODEL	**CFAH1601A-YYB-JP**	
APPROVAL	**BY:**	**DATA:**

SALES BY	APPROVED BY	CHECKED BY	PREPARED BY

Crystalfontz America, Inc.
15611 East Washington Road
Valleyford, WA 99036-9747
Phone: (509) 291-3514
Fax: (509) 291-3345
e-mail: sales@crystalfontz.com
http://www.crystalfontz.com

Crystalfontz America, Inc.
15611 East Washington Road
Valleyford, WA 99036
Phone: (509) 291-3514
Fax: (509) 291-3345
http://www.crystalfontz.com
email: sales@crystalfontz.com

Contents

Crystalfontz America, Inc.
15611 East Washington Road
Valleyford, WA 99036
Phone: (509) 291-3514
Fax: (509) 291-3345
http://www.crystalfontz.com
email: sales@crystalfontz.com

1.Module Classification Information

CFA H 1601 A－Y Y B－ JP
① ② ③ ④ ⑤ ⑥ ⑦ ⑧

①	Brand: CRYSTALFONTZ AMERICA, INCORPORATED		
②	Display Type: **H➔Character Type**, G➔Graphic Type, X➔TAB		
③	Display's logical dimensions: **16 characters, 1 line**		
④	Model serials no.		
⑤	Backlight Type:	N➔ Without backlight B➔ EL, Blue green D➔ EL, Green W➔ EL, White F➔ CCFL, White **Y➔ LED, Yellow Green**	A➔ LED, Amber R➔ LED, Red O➔ LED, Orange G➔ LED, Green
⑥	LCD Mode:	B➔ TN Positive, Gray N➔ TN Negative, G➔ STN Positive, Gray **Y➔ STN Positive, Yellow Green** M➔ STN Negative, Blue F➔ FSTN Positive	T➔ FSTN Negative
⑦	LCD Polarizer Type/ Temperature range/ View direction	A➔ Reflective, N.T, 6:00 D➔ Reflective, N.T, 12:00 G➔ Reflective, W. T, 6:00 J➔ Reflective, W. T, 12:00 **B➔ Transflective, N.T,6:00** E➔ Transflective, N.T.12:00	H➔ Transflective, W.T,6:00 K➔ Transflective, W.T,12:00 C➔ Transmissive, N.T,6:00 F➔ Transmissive, N.T,12:00 I➔ Transmissive, W. T, 6:00 L➔ Transmissive, W.T,12:00
⑧	Special Code:	**JP : English and Japanese standard font**	

2.Precautions in use of LCD Modules

(1)Avoid applying excessive shocks to the module or making any alterations or modifications to it.

(2)Don't make extra holes on the printed circuit board, modify its shape or change the components of LCD module.

(3)Don't disassemble the LCM.

(4)Don't operate it above the absolute maximum rating.

(5)Don't drop, bend or twist LCM.

(6)Soldering: only to the I/O terminals.

(7)Storage: please storage in anti-static electricity container and clean environment.

3.General Specification

Item	Dimension	Unit
Number of Characters	16 characters x 1 Lines	—
Module dimension	80.0 x 36.0 x 13.2(MAX)	mm
View area	66.0 x 16.0	mm
Active area	59.62 x 6.56	mm
Dot size	0.55 x 0.75	mm
Dot pitch	0.63 x 0.83	mm
Character size	3.07 x 6.56	mm
Character pitch	3.77 x 6.56	mm
LCD type	STN, Positive, Transflective, Yellow Green	
Duty	1/16	
View direction	6 o'clock	
Backlight Type	LED Yellow Green	

4.Absolute Maximum Ratings

Item	Symbol	Min	Typ	Max	Unit
Operating Temperature	T_{OP}	0	—	+50	℃
Storage Temperature	T_{ST}	-10	—	+60	℃
Input Voltage	V_I	V_{SS}	—	V_{DD}	V
Supply Voltage For Logic	V_{DD}-V_{SS}	-0.3	—	7	V
Supply Voltage For LCD	V_{DD}-V_0	-0.3	—	13	V

5.Electrical Characteristics

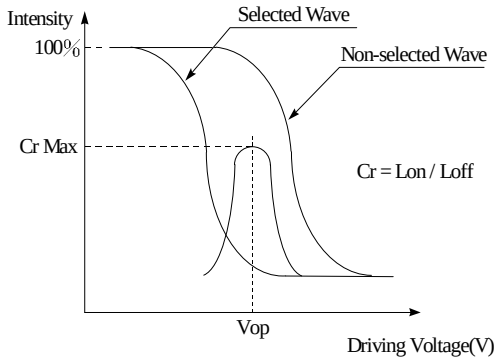
Item	Symbol	Condition	Min	Typ	Max	Unit
Supply Voltage For Logic	V_{DD}-V_{SS}	—	4.5	—	5.5	V
Supply Voltage For LCD	V_{DD}-V_0	Ta=0℃	—	—	4.8	V
		Ta=25℃	—	4.5	—	V
		Ta=50℃	4.2	—	—	V
Input High Volt.	V_{IH}	—	2.2	—	V_{DD}	V
Input Low Volt.	V_{IL}	—	—	—	0.6	V
Output High Volt.	V_{OH}	—	2.4	—	—	V
Output Low Volt.	V_{OL}	—	—	—	0.4	V
Supply Current	I_{DD}	V_{DD}=5V	—	1.2	—	mA

Crystalfontz America, Inc.
15611 East Washington Road
Valleyford, WA 99036
Phone: (509) 291-3514
Fax: (509) 291-3345
http://www.crystalfontz.com
email: sales@crystalfontz.com

6.Optical Characteristics

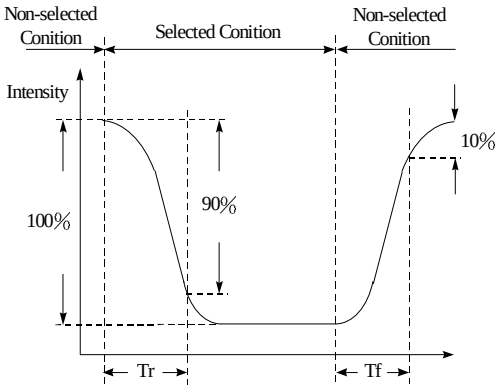
Item	Symbol	Condition	Min	Typ	Max	Unit
View Angle	(V) θ	CR≧2	10	—	105	deg
	(H) φ	CR≧2	-30	—	30	deg
Contrast Ratio	CR	—	—	3	—	—
Response Time	T rise	—	—	150	200	ms
	T fall	—	—	150	200	ms

Definition of Operation Voltage (Vop)



[positive type]

Definition of Response Time (Tr , Tf)



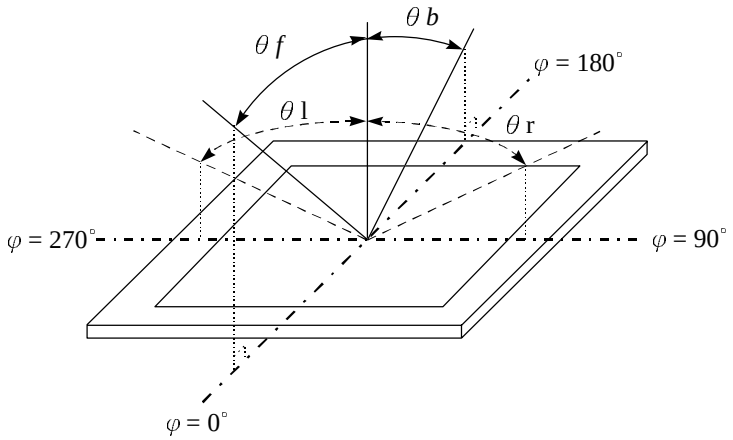
[positive type]

Conditions :

Operating Voltage : Vop Viewing Angle(θ , φ) : 0° , 0°

Frame Frequency : 64 HZ Driving Waveform : 1/N duty , 1/a bias

Definition of viewing angle(CR≧2)



7.Interface Pin Function

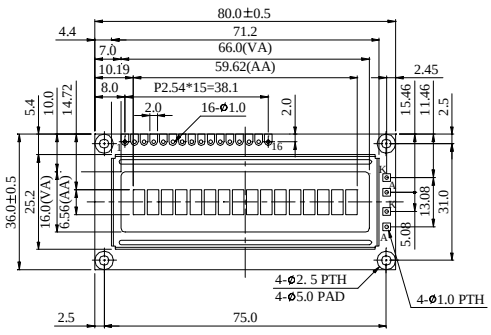
Pin No.	Symbol	Level	Description
1	V_{SS}	0V	Ground
2	V_{DD}	5.0V	Supply Voltage for logic
3	VO	(Variable)	Operating voltage for LCD
4	RS	H/L	H: DATA, L: Instruction code
5	R/W	H/L	H: Read(MPU→Module) L: Write(MPU→Module)
6	E	H,H→L	Chip enable signal
7	DB0	H/L	Data bit 0
8	DB1	H/L	Data bit 1
9	DB2	H/L	Data bit 2
10	DB3	H/L	Data bit 3
11	DB4	H/L	Data bit 4
12	DB5	H/L	Data bit 5
13	DB6	H/L	Data bit 6
14	DB7	H/L	Data bit 7
15	A	—	LED +
16	K	—	LED—

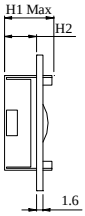
Crystalfontz America, Inc.
15611 East Washington Road
Valleyford, WA 99036

Phone: (509) 291-3514
Fax: (509) 291-3345

http://www.crystalfontz.com
email: sales@crystalfontz.com

8.Contour Drawing &Block Diagram



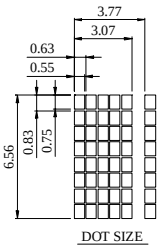


LED B/L

LED-H/L B/L

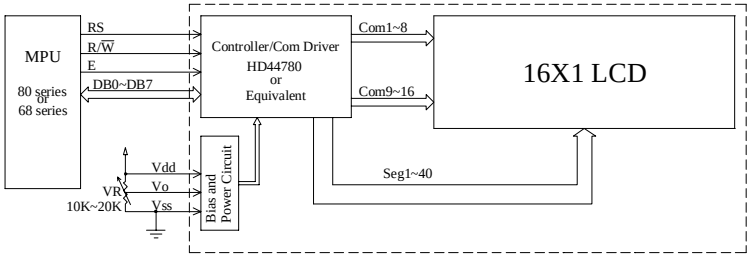
	High	Low
H1	13.2	
H2	8.6	

PIN NO.	SYMBOL
1	Vss
2	Vdd
3	Vo
4	RS
5	R/$\overline{W}$
6	E
7	DB0
8	DB1
9	DB2
10	DB3
11	DB4
12	DB5
13	DB6
14	DB7
15	A/Vee
16	K



DOT SIZE

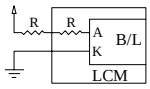
The non-specified tolerance of dimension is ±0.3mm.



External contrast adjustment.

LED B/L Drive Method

Drive from pin15, pin16



Character located	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
DDRAM address	00	01	02	03	04	05	06	07	40	41	42	43	44	45	46	47

2-line display mode.

9.Function Description

The LCD display Module is built in a LSI controller, the controller has two 8-bit registers, an instruction register (IR) and a data register (DR).
The IR stores instruction codes, such as display clear and cursor shift, and address information for display data RAM (DDRAM) and character generator (CGRAM). The IR can only be written from the MPU. The DR temporarily stores data to be written or read from DDRAM or CGRAM. When address information is written into the IR, then data is stored into the DR from DDRAM or CGRAM. By the register selector (RS) signal, these two registers can be selected.

RS	R/W	Operation
0	0	IR write as an internal operation (display clear, etc.)
0	1	Read busy flag (DB7) and address counter (DB0 to DB7)
1	0	Write data to DDRAM or CGRAM (DR to DDRAM or CGRAM)
1	1	Read data from DDRAM or CGRAM (DDRAM or CGRAM to DR)

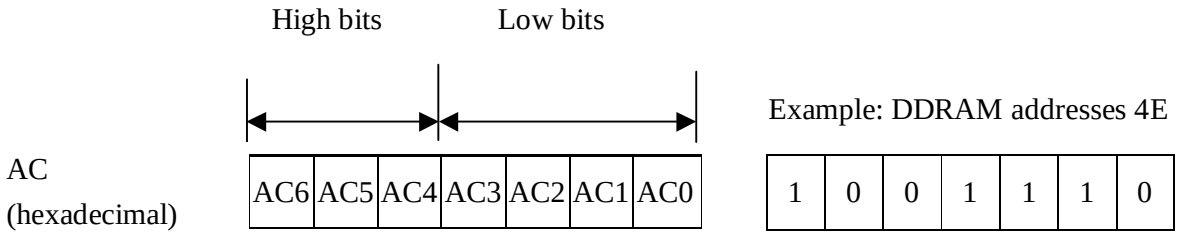
Busy Flag (BF)
When the busy flag is 1, the controller LSI is in the internal operation mode, and the next instruction will not be accepted. When RS=0 and R/W=1, the busy flag is output to DB7. The next instruction must be written after ensuring that the busy flag is 0.

Address Counter (AC)
The address counter (AC) assigns addresses to both DDRAM and CGRAM

Crystalfontz America, Inc.
15611 East Washington Road
Valleyford, WA 99036
Phone: (509) 291-3514
Fax: (509) 291-3345
http://www.crystalfontz.com
email: sales@crystalfontz.com

Display Data RAM (DDRAM)

This DDRAM is used to store the display data represented in 8-bit character codes. Its extended capacity is 80×8 bits or 80 characters. Below figure is the relationships between DDRAM addresses and positions on the liquid crystal display.



Display position DDRAM address

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
00	01	02	03	04	05	06	07	40	41	42	43	44	45	46	47

1-Line by 16-Character Display

Character Generator ROM (CGROM)

The CGROM generate 5×8 dot or 5×10 dot character patterns from 8-bit character codes. See Table 2.

Character Generator RAM (CGRAM)

In CGRAM, the user can rewrite character by program. For 5×8 dots, eight character patterns can be written, and for 5×10 dots, four character patterns can be written.

Write into DDRAM the character code at the addresses shown as the left column of table 1. To show the character patterns stored in CGRAM.

Crystalfontz America, Inc.
15611 East Washington Road
Valleyford, WA 99036

Phone: (509) 291-3514
Fax: (509) 291-3345

http://www.crystalfontz.com
email: sales@crystalfontz.com

Relationship between CGRAM Addresses, Character Codes (DDRAM) and Character patterns

Table 1.

For 5 * 8 dot character patterns

Character Codes (DDRAM data)	CGRAM Address	Character Patterns (CGRAM data)	
7 6 5 4 3 2 1 0	5 4 3 2 1 0	7 6 5 4 3 2 1 0	
High Low	High Low	High Low	
0 0 0 0 * 0 0 0	0 0 0	0 0 0	Character pattern(1)
		1 1 1	Cursor pattern
0 0 0 0 * 0 0 1	0 0 1	0 0 0	Character pattern(2)
		1 1 1	Cursor pattern
		0 0 0 0 0 1	
0 0 0 0 * 1 1 1	1 1 1	1 0 0 1 0 1 1 1 0 1 1 1	

For 5 * 10 dot character patterns

Character Codes (DDRAM data)	CGRAM Address	Character Patterns (CGRAM data)	
7 6 5 4 3 2 1 0	5 4 3 2 1 0	7 6 5 4 3 2 1 0	
High Low	High Low	High Low	
0 0 0 0 * 0 0 0	0 0	0 0 0 0 0	Character pattern
		1 0 1 0	Cursor pattern
		1 1 1 1	

■ : " High "

Crystalfontz America, Inc.
15611 East Washington Road
Valleyford, WA 99036

Phone: (509) 291-3514
Fax: (509) 291-3345

http://www.crystalfontz.com
email: sales@crystalfontz.com

10.Character Generator ROM Pattern

Table.2

Upper 4 bit / Lower 4 bit	LLLL	LLLH	LLHL	LLHH	LHLL	LHLH	LHHL	LHHH	HLLL	HLLH	HLHL	HLHH	HHLL	HHLH	HHHL	HHHH
LLLL	CG RAM (1)															
LLLH	(2)															
LLHL	(3)															
LLHH	(4)															
LHLL	(5)															
LHLH	(6)															
LHHL	(7)															
LHHH	(8)															
HLLL	(1)															
HLLH	(2)															
HLHL	(3)															
HLHH	(4)															
HHLL	(5)															
HHLH	(6)															
HHHL	(7)															
HHHH	(8)															

Crystalfontz America, Inc.
15611 East Washington Road
Valleyford, WA 99036
Phone: (509) 291-3514
Fax: (509) 291-3345
http://www.crystalfontz.com
email: sales@crystalfontz.com

11.Instruction Table

Instruction	Instruction Code										Description	Execution time (fosc=270Khz)
	RS	R/W	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0		
Clear Display	0	0	0	0	0	0	0	0	0	1	Write “00H” to DDRAM and set DDRAM address to “00H” from AC	1.53ms
Return Home	0	0	0	0	0	0	0	0	1	—	Set DDRAM address to “00H” from AC and return cursor to its original position if shifted. The contents of DDRAM are not changed.	1.53ms
Entry Mode Set	0	0	0	0	0	0	0	1	I/D	SH	Assign cursor moving direction and enable the shift of entire display.	39μs
Display ON/OFF Control	0	0	0	0	0	0	1	D	C	B	Set display (D), cursor (C), and blinking of cursor (B) on/off control bit.	39μs
Cursor or Display Shift	0	0	0	0	0	1	S/C	R/L	—	—	Set cursor moving and display shift control bit, and the direction, without changing of DDRAM data.	39μs
Function Set	0	0	0	0	1	DL	N	F	—	—	Set interface data length (DL:8-bit/4-bit), numbers of display line (N:2-line/1-line)and, display font type (F:5×11 dots/5×8 dots)	39μs
Set CGRAM Address	0	0	0	1	AC5	AC4	AC3	AC2	AC1	AC0	Set CGRAM address in address counter.	39μs
Set DDRAM Address	0	0	1	AC6	AC5	AC4	AC3	AC2	AC1	AC0	Set DDRAM address in address counter.	39μs
Read Busy Flag and Address	0	1	BF	AC6	AC5	AC4	AC3	AC2	AC1	AC0	Whether during internal operation or not can be known by reading BF. The contents of address counter can also be read.	0μs
Write Data to RAM	1	0	D7	D6	D5	D4	D3	D2	D1	D0	Write data into internal RAM (DDRAM/CGRAM).	43μs
Read Data from RAM	1	1	D7	D6	D5	D4	D3	D2	D1	D0	Read data from internal RAM (DDRAM/CGRAM).	43μs

＊ ”—”：don’t care

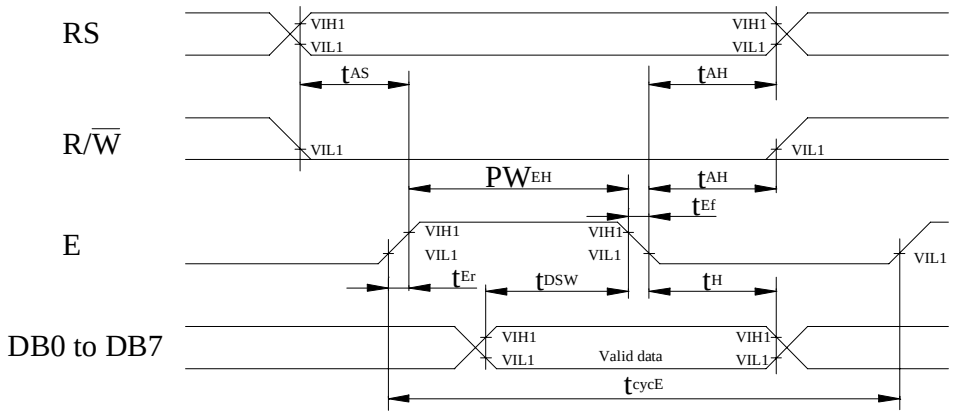
Crystalfontz America, Inc.
15611 East Washington Road
Valleyford, WA 99036

Phone: (509) 291-3514
Fax: (509) 291-3345

http://www.crystalfontz.com
email: sales@crystalfontz.com

12.Timing Characteristics

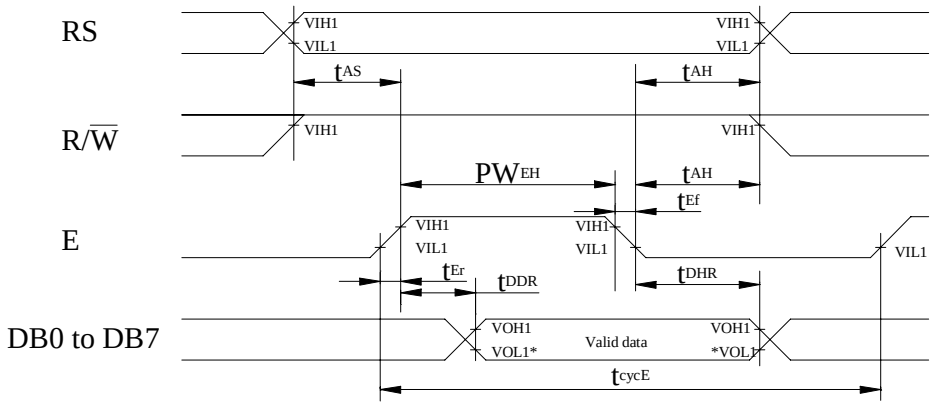
12.1 Write Operation



Ta=25℃, VDD=5.0± 0.5V

Item	Symbol	Min	Typ	Max	Unit
Enable cycle time	t_{cycE}	400	—	—	ns
Enable pulse width (high level)	PW_{EH}	150	—	—	ns
Enable rise/fall time	t_{Er},t_{Ef}	—	—	25	ns
Address set-up time (RS, R/W to E)	t_{AS}	30	—	—	ns
Address hold time	t_{AH}	10	—	—	ns
Data set-up time	t_{DSW}	40	—	—	ns
Data hold time	t_H	10	—	—	ns

12.2 Read Operation



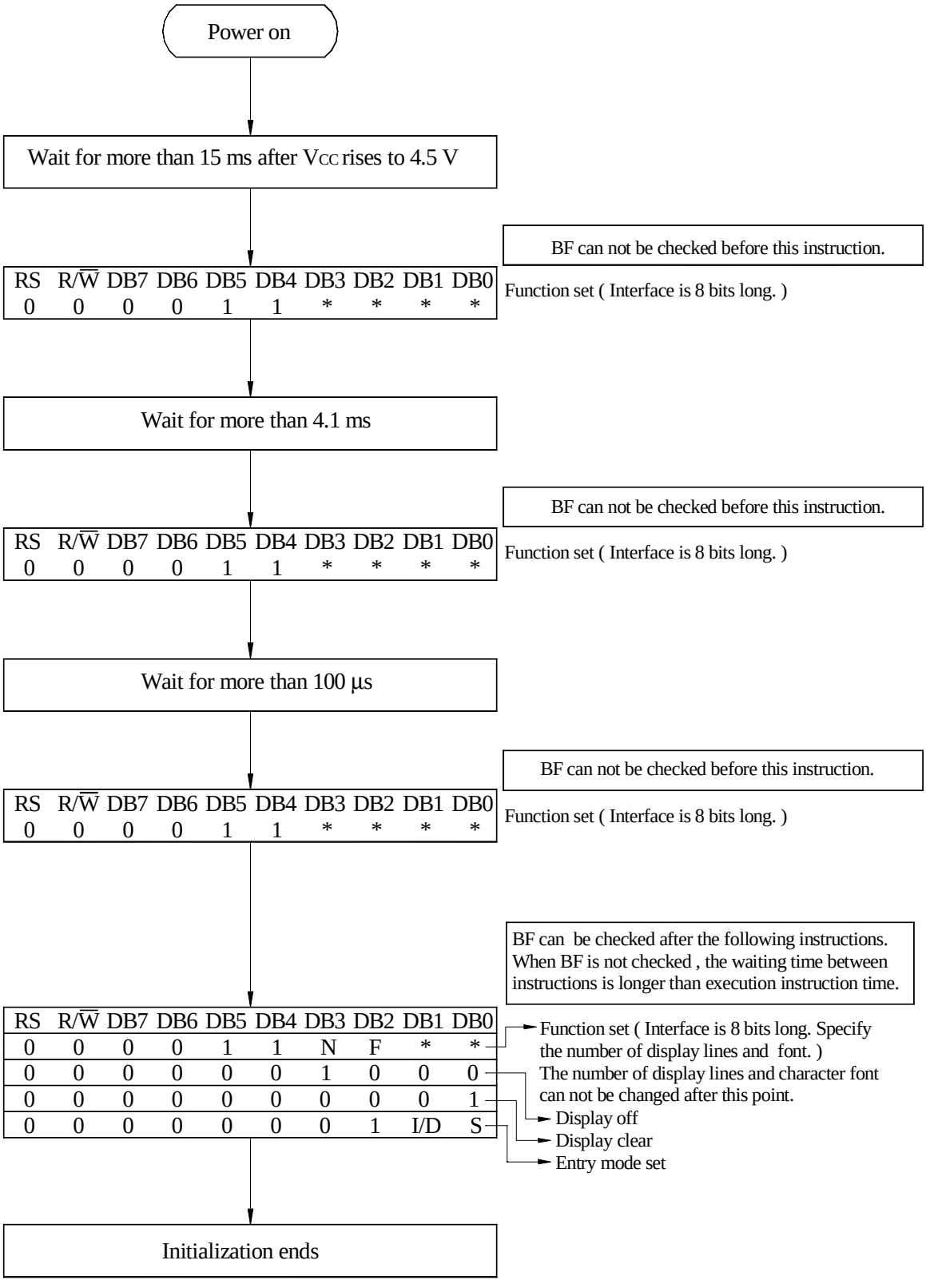
NOTE: *VOL1 is assumed to be 0.8V at 2 MHZ operation.

Ta=25℃, VDD=5.0± 0.5V

Item	Symbol	Min	Typ	Max	Unit
Enable cycle time	t_{cycE}	400	—	—	ns
Enable pulse width (high level)	PW_{EH}	150	—	—	ns
Enable rise/fall time	t_{Er},t_{Ef}	—	—	25	ns
Address set-up time (RS, R/W to E)	t_{AS}	30	—	—	ns
Address hold time	t_{AH}	10	—	—	ns
Data delay time	t_{DDR}	—	—	100	ns
Data hold time	t_{DHR}	20	—	—	ns

Crystalfontz America, Inc.
15611 East Washington Road
Valleyford, WA 99036
Phone: (509) 291-3514
Fax: (509) 291-3345
http://www.crystalfontz.com
email: sales@crystalfontz.com

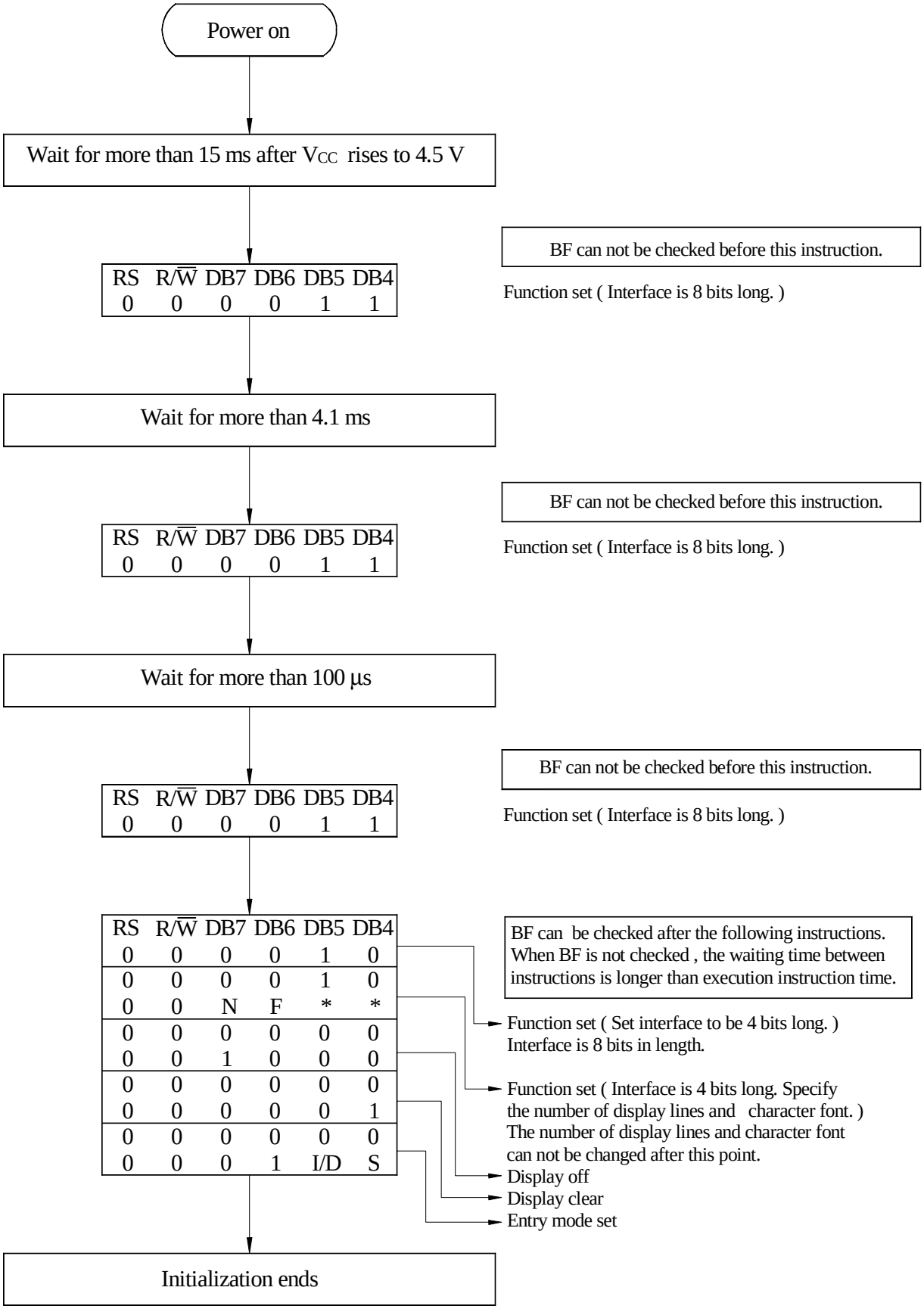
13.Initializing of LCM



Crystalfontz America, Inc.
15611 East Washington Road
Valleyford, WA 99036

Phone: (509) 291-3514
Fax: (509) 291-3345

http://www.crystalfontz.com
email: sales@crystalfontz.com



4-Bit Ineterface

Crystalfontz America, Inc.
15611 East Washington Road
Valleyford, WA 99036
Phone: (509) 291-3514
Fax: (509) 291-3345
http://www.crystalfontz.com
email: sales@crystalfontz.com

14.Quality Assurance

Screen Cosmetic Criteria

Item	Defect	Judgment Criterion	Partition
1	Spots	A)Clear Size: d mm — Acceptable Qty in active area d ≦0.1 — Disregard 0.1<d≦0.2 — 6 0.2<d≦0.3 — 2 0.3<d — 0 Note: Including pin holes and defective dots which must be within one pixel size. B)Unclear Size: d mm — Acceptable Qty in active area d ≦0.2 — Disregard 0.2<d≦0.5 — 6 0.5<d≦0.7 — 2 0.7<d — 0	Minor
2	Bubbles in Polarizer	Size: d mm — Acceptable Qty in active area d≦0.3 — Disregard 0.3<d≦1.0 — 3 1.0<d≦1.5 — 1 1.5<d — 0	Minor
3	Scratch	In accordance with spots cosmetic criteria. When the light reflects on the panel surface, the scratches are not to be remarkable.	Minor
4	Allowable Density	Above defects should be separated more than 30mm each other.	Minor
5	Coloration	Not to be noticeable coloration in the viewing area of the LCD panels. Back-light type should be judged with back-light on state only.	Minor

15.Reliability

Content of Reliability Test

Environmental Test			
Test Item	**Content of Test**	**Test Condition**	**Applicable Standard**
High Temperature storage	Endurance test applying the high storage temperature for a long time.	60℃ 200hrs	——
Low Temperature storage	Endurance test applying the high storage temperature for a long time.	-10℃ 200hrs	——
High Temperature Operation	Endurance test applying the electric stress (Voltage & Current) and the thermal stress to the element for a long time.	50℃ 200hrs	——
Low Temperature Operation	Endurance test applying the electric stress under low temperature for a long time.	0℃ 200hrs	——
High Temperature/ Humidity Storage	Endurance test applying the high temperature and high humidity storage for a long time.	60℃,90%RH 96hrs	——
High Temperature/ Humidity Operation	Endurance test applying the electric stress (Voltage & Current) and temperature / humidity stress to the element for a long time.	50℃,90%RH 96hrs	——
Temperature Cycle	Endurance test applying the low and high temperature cycle. -10℃ ← 25℃ → 60℃ 30min 5min 30min 1 cycle	-10℃/60℃ 10 cycles	——
Mechanical Test			
Vibration test	Endurance test applying the vibration during transportation and using.	10~22Hz→1.5mmp-p 22~500Hz→1.5G Total 0.5hrs	——
Shock test	Constructional and mechanical endurance test applying the shock during transportation.	50G Half sign wave 11 msedc 3 times of each direction	——
Atmospheric pressure test	Endurance test applying the atmospheric pressure during transportation by air.	115mbar 40hrs	——
Others			
Static electricity test	Endurance test applying the electric stress to the terminal.	VS=800V,RS=1.5kΩ CS=100pF 1 time	——

***Supply voltage for logic system=5V. Supply voltage for LCD system =Operating voltage at 25℃

16.Backlight Information

Specification

PARAMETER	SYMBOL	MIN	TYP	MAX	UNIT	TEST CONDITION
Supply Current	**ILED**	—	**130**	—	**mA**	**V=4.2V**
Supply Voltage	**V**	—	**4.2**	**4.6**	**V**	—
Reverse Voltage	**VR**	—	—	**8**	**V**	—
Luminous Intensity	**IV**	**60**	—	—	**CD/M^2**	**ILED=130mA**
Wave Length	**λp**	—	**571**	—	**nm**	**ILED=130mA**
Life Time	—	—	**100000**	—	**Hr.**	**V≦4.6V**
Color	**Yellow Green**					